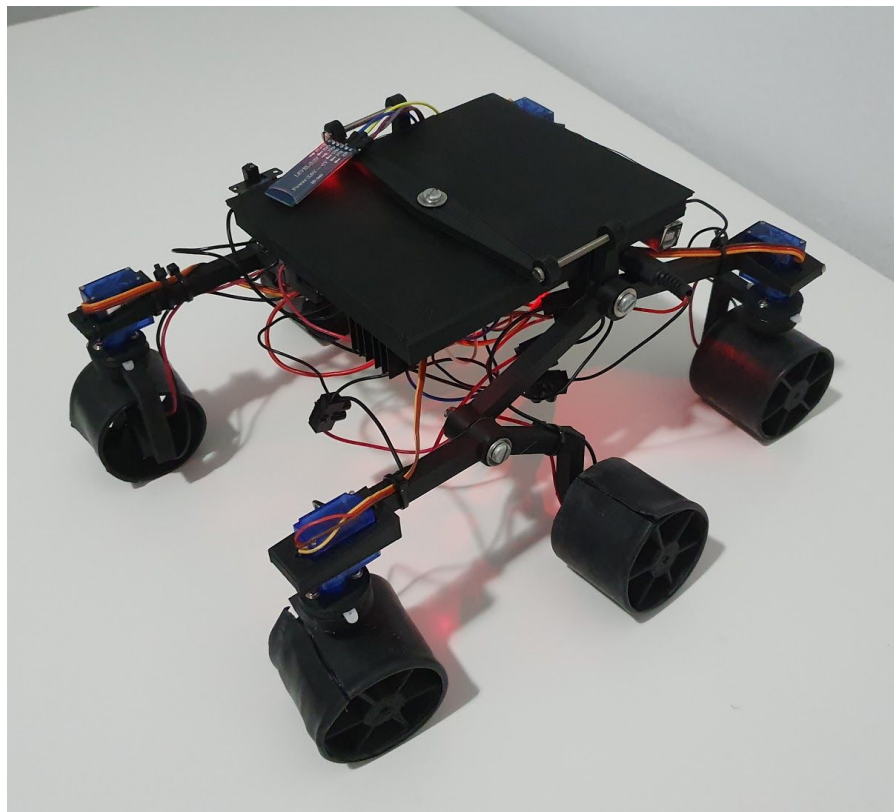


Memoria del Voidcuriosity



Daniel Rodríguez Caro
Alejandro Lauriño Ruiz
Raúl Gimeno Cervera

1ºBachillerato A
IES Vicente Aleixandre
//2019

Índice

Índice	1
1. Finalidad del Sistema	2
2. Hardware	2
3. Software	4
4. Funcionamiento	6
5. Evaluación	6
6. Bibliografía y webgrafía	7

1. Finalidad del Sistema

El proyecto es una reproducción a escala 1:10 del astromóvil lanzado en la misión Mars Science Laboratory.

En un principio, enviamos las órdenes para mover los servos y hacer girar los motores a través del monitor serie, con el robot conectado al ordenador (pmv).

Cuando ya teníamos eso, pasamos a usar un módulo bluetooth y App Inventor para controlarlo remotamente desde un dispositivo móvil.

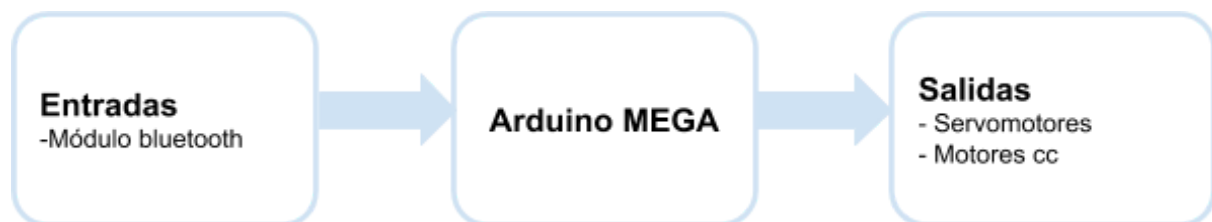
El sistema consta de 6 motores cc motrices (N20 dc motors) y 4 servomotores directrices (9g servo), además del mecanismo rocker-boogie que le permitirá superar obstáculos no muy complejos con facilidad. Gracias a este sistema y la estabilidad del móvil, podrá desenvolverse sin dificultades en superficies escarpadas y terrenos complejos.

2. Búsqueda de información

La mayor parte de la información ha sido conseguida en el [repositorio del proyecto](#) que se encuentra en la web de Bricolabs y en los Github's de [@felixstpd](#) y [@javacasm](#) los cuáles han sido una ayuda fundamental a la hora de realizar el proyecto.

3. Hardware

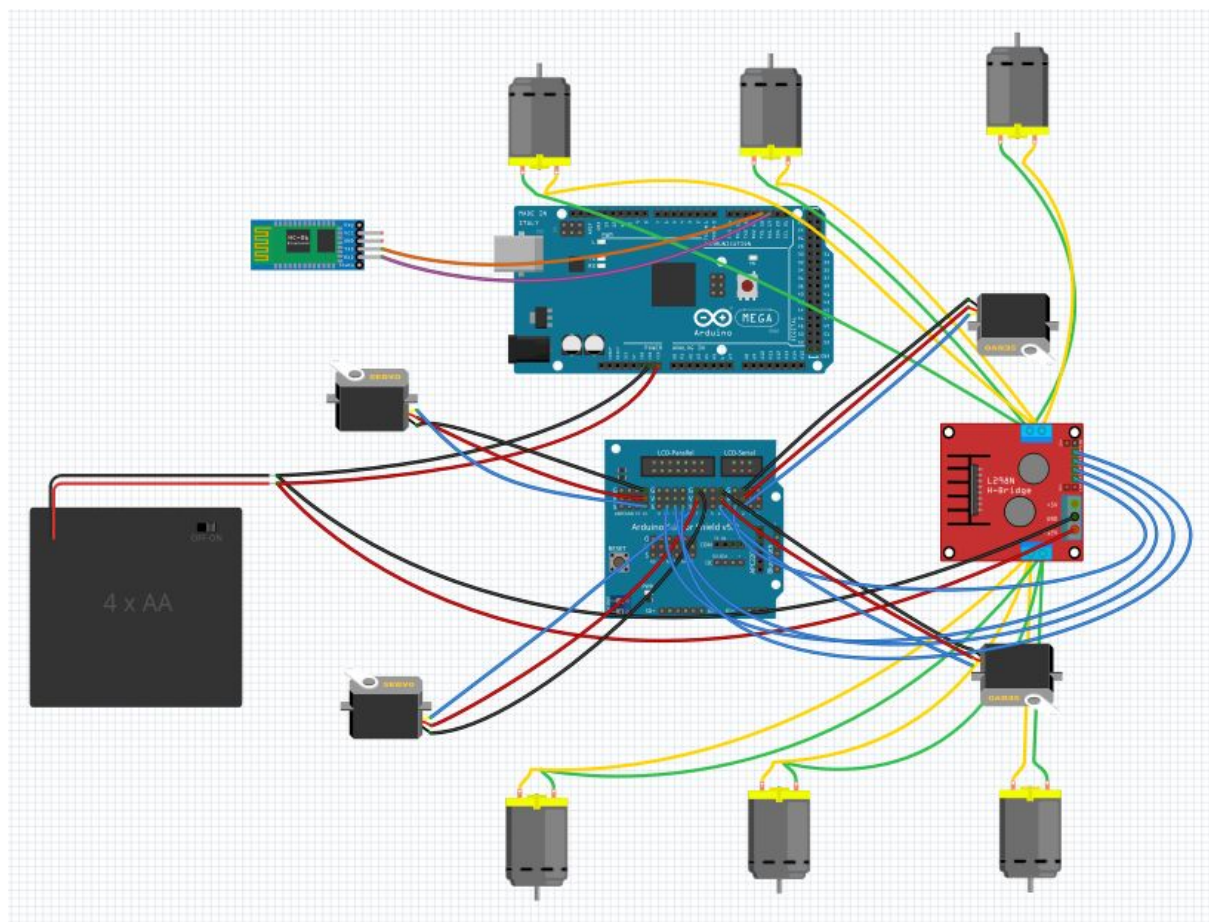
a. Esquema de entradas y salidas



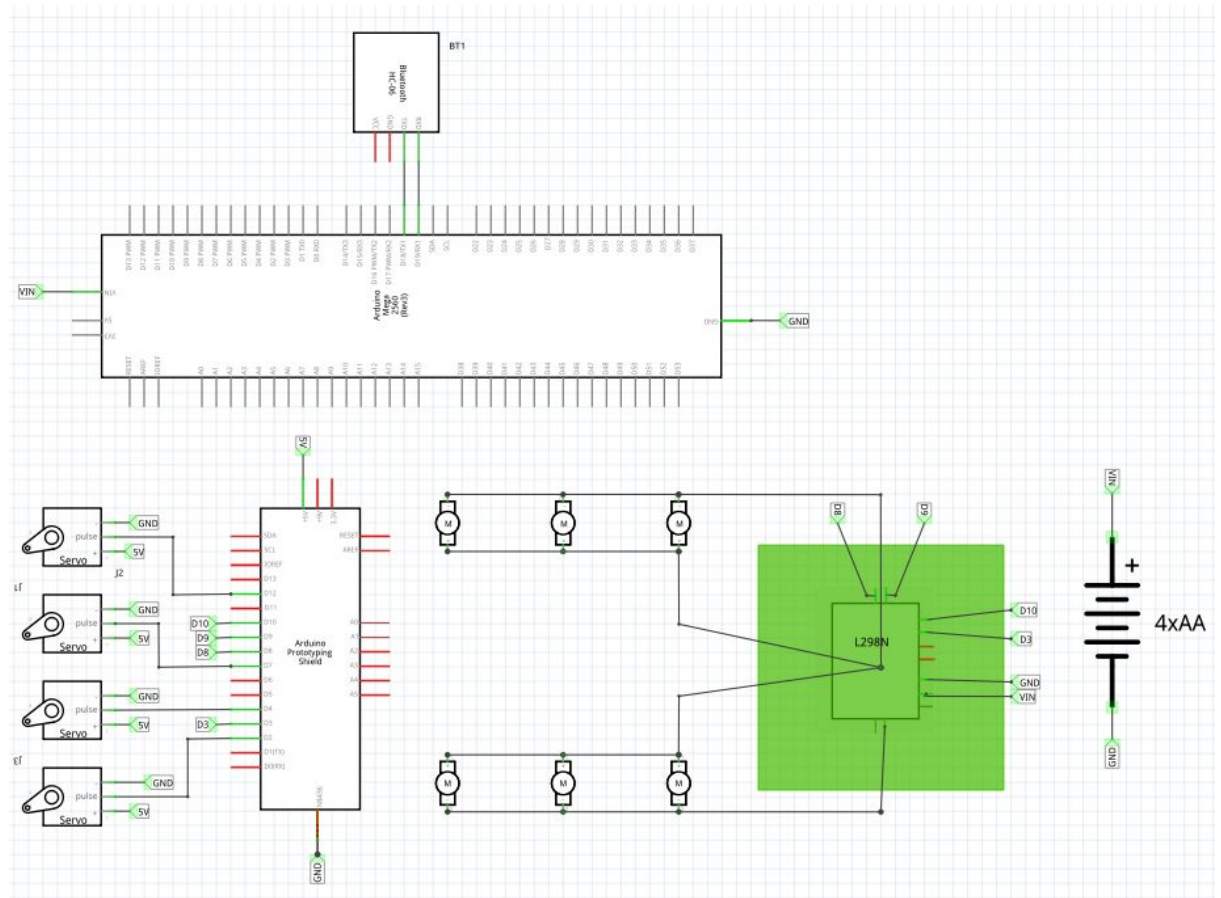
b. Lista de materiales

Material	Cantidad
Placa Arduino MEGA	1
Arduino sensor shield v5.0	1
L298	1
N20 dc motors	6
9g servos	4
18650 baterías	2
Tornillería	General

c. Esquema de la protoboard



d. Esquema electrónico



4. Software

```
/* Código v3.0
Proyecto Void Curiosity
Danny Caro
Alejandro Lauriño
Raúl Gimeno
1º BTO TIND-TIC
Basado en código de @javacasm
IES Vicente Aleixandre */

#include <Servo.h> //incluimos la librería para poder controlar los servomotores

//Definimos los pines de los servomotores
#define S1 12
#define S2 7
#define S3 4
#define S4 2
Servo s1, s2, s3, s4;

// Pines de control de los motores
// Pines motor derecho
const int inA = 3;
const int inB = 10;
const int enA = 5;

// Pines motor izquierdo
const int inC = 9;
const int inD = 8;
const int enB = 6;

int value = 0; // variable para almacenar el valor numerico
int posS1, posS2, posS3, posS4; //Creamos las variables para almacenar la posición de los
servos
int pasoServo = 5; // Incremento de posicion de los servos
int pwmValue = 0; // variable para enviar el codigo pwm al led

//Creamos una función para definir la posición inicial de los servos
void setServosInitialState() {
    posS1 = 97;
    posS2 = 105;
    posS3 = 98;
    posS4 = 82;
}

// La velocidad va desde -255 a 255, con valores negativos moviendose hacia atras
```

```
int velocidad = 0;
int pasoVelocidad = 10; // Usaremos este incremento

// Nos permite dar velocidad a cada motor
void setSpeed(int enB, int speed) {
  if (speed > 0) {
    analogWrite(enB, speed);
  }
  else {
    analogWrite(enB, - speed);
  }
}

//Establecemos posición de los servos y la velocidad de los motores
void setState() {
  setSpeed(enB, velocidad);
  setSpeed(enA, velocidad);
  s1.write(posS1);
  s2.write(posS2);
  s3.write(posS3);
  s4.write(posS4);
}

// El motor gira en sentido horario
void clockWise() {
  digitalWrite(inA, LOW);
  digitalWrite(inB, HIGH);
  digitalWrite(inC, LOW);
  digitalWrite(inD, HIGH);
}

// Función para que el motor gire en sentido antihorario
void antiClockWise() {
  digitalWrite(inA, HIGH);
  digitalWrite(inB, LOW);
  digitalWrite(inC, HIGH);
  digitalWrite(inD, LOW);
}

void setup() {
  // Configuramos los pines de los motores como salida
  pinMode(inA, OUTPUT);
  pinMode(inB, OUTPUT);
  pinMode(inC, OUTPUT);
  pinMode(inD, OUTPUT);
}
```

```
// Configuramos los pines de los servos como salida
pinMode(S1, OUTPUT);
pinMode(S2, OUTPUT);
pinMode(S3, OUTPUT);
pinMode(S4, OUTPUT);

// Configuramos las comunicaciones
Serial1.begin(9600);

// Configuramos los servos
s1.attach(S1);
s2.attach(S2);
s3.attach(S3);
s4.attach(S4);
setServosInitialState();
setState();
}

void loop() {
  /*
    Control de velocidad y giro por movimiento
    F el robot avanza
    S detiene en seco
    B el robot retrocede

    Control de giro usando los servos
    L girar a la izquierda
    R girar a la derecha
    O servos rectos
  */

  if (Serial1.available() > 0) {
    char caracter = Serial1.read();
    if (caracter >= '0' && caracter <= '9') {
      //Acumula los datos numericos multiplicando por 10 el valor acumulado
      value = (value * 10) + (caracter - '0'); // Resta 48 que es el valor decimal del 0 ASCII
    }
    else if (caracter == '>') // uso > como finalizador
    {
      pwmValue = value; // Guarda el valor en la variable pwmValue
      value = 0; // Dejamos lista la variable para volver a escribir en ella
    }

    else {
      switch (caracter) {
```



```
case 'F': //Utilizamos para que el robot pueda caminar hacia delante
    clockWise();
    velocidad = pwmValue;
    break;
case 'B': //Utilizamos para que el robot pueda dar marcha atrás
    antiClockWise();
    velocidad = pwmValue;
    break;
case 'S': //Paramos los motores
    velocidad = 0;
    break;
case 'L': // Mas giro a la izquierda
    posS1 -= pasoServo;
    posS4 -= pasoServo;
    posS3 += pasoServo;
    posS2 += pasoServo;
    break;
case 'O': // Ponemos los servomotores en la posición inicial
    setServosInitialState();
    break;
case 'R': // Mas giro a la derecha
    posS1 += pasoServo;
    posS4 += pasoServo;
    posS3 -= pasoServo;
    posS2 -= pasoServo;
    break;
}
}
// Vemos si nos hemos pasado de posiciones
if (posS1 < 0) posS1 = 0;
if (posS2 < 0) posS2 = 0;
if (posS3 < 0) posS3 = 0;
if (posS4 < 0) posS4 = 0;
if (posS1 > 180) posS1 = 180;
if (posS2 > 180) posS2 = 180;
if (posS3 > 180) posS3 = 180;
if (posS4 > 180) posS4 = 180;

// Vemos si nos pasamos en las velocidades
if (velocidad > 255 ) velocidad = 255;
if (velocidad < -255 ) velocidad = -255;

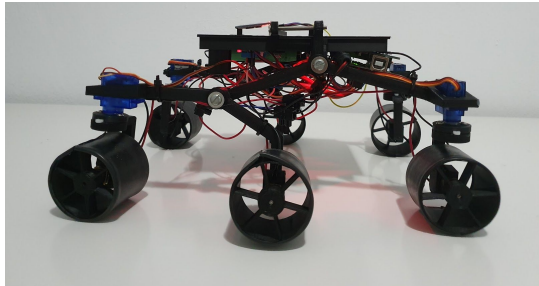
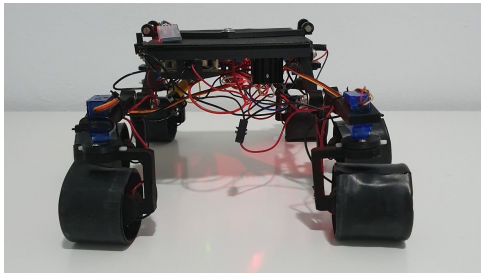
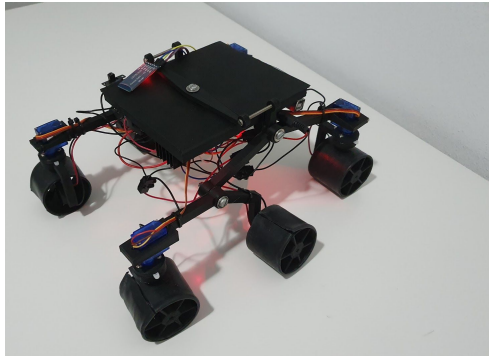
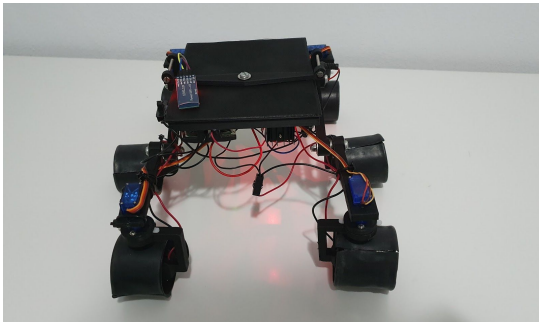
// Establecemos las posiciones de los servos y la velocidad de los motores
setState();
}
```

5. *Funcionamiento*

a. Vídeos

[Enlace al vídeo](#)

b. Fotos

Vista de perfil	
Alzado	
Vista diagonal	
Vista de planta	

6. Evaluación

a. Qué funciona bien y qué se puede mejorar.

- **Sistema rocker-bogie:** El sistema funciona bien y consigue superar obstáculos de una altura del diámetro de la rueda más o menos. Podríamos mejorarlo haciendo que pase unos obstáculos de un poco más de inclinación.

b. Problemas que hemos tenido y soluciones.

- **Cortocircuito:** Tuvimos un cortocircuito al poner las pilas en el portapilas y entonces este se quemó. Solución: Revisamos todas las conexiones y utilizamos una fuente de alimentación.
- **Pieza simétrica:** Cuando hicimos la impresión 3D no hicimos la pieza simétrica de una de las patas, por lo que las ruedas quedaban mal posicionadas. Solución: Tuvimos que repetir la pieza haciéndola simétrica.
- **Falta de pines PWM:** Al comenzar el proyecto, en el hardware usábamos Arduino UNO pero nos faltaban pines PWM. Solución: Cambiamos Arduino UNO por el Arduino MEGA que tiene más pines PWM.
- **Alimentación:** No hemos tenido información de cómo alimentar el Arduino SensorShield v5.0. Solución: Hemos conectado la alimentación al JACK del Arduino Mega.

c. Propuestas para la mejora y ampliación del proyecto.

- **Wi-Fi:** Podríamos controlar el proyecto mediante Wi-Fi, y así poder manejarlo por internet.
- **Cámara:** Podríamos añadir una cámara al proyecto con la que podríamos ver por donde está en una pantalla.
- **Sensor GPS:** Al añadir un sensor de ubicación podríamos saber el lugar en el que se encuentra en cada momento.

7. Bibliografía y webgrafía

- https://github.com/javacasm/curiosity_bt
- https://bricolabs.cc/wiki/proyectos/curiosity_bt
- Twitter: @felixstdp @javacasm