TipFinder

GRUPO 11

Jorge García Cantúa
Nuno Durao Moreira Vicente
Carlos Villalba Castillejo

Sistemas empotrados y tiempo real
25/04/2018

ÍNDICE

1-Introducción. 2
2-Desarrollo del proyecto. 3
3-Presupuesto. 6
4-Implementación 7
4.1-Circuito controlador del coche 7
4.2-Circuito detector de metales y GPS 13
5-Problemas encontrados y soluciones. 19
6-Posibles mejoras. 20

1-Introducción.

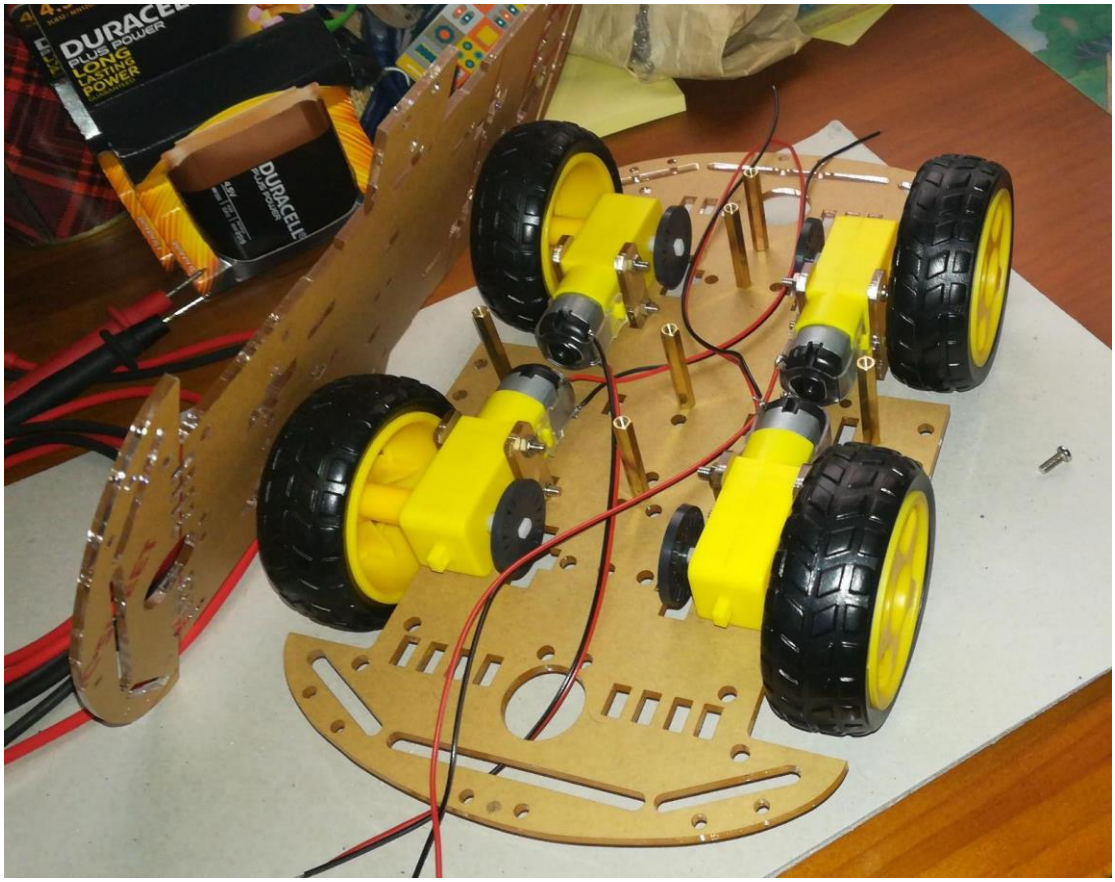
Hemos construido un coche que controlamos mediante el teléfono móvil y que detecta metales. Cuando detecta el metal, se envía información sobre la posición de dicho metal (longitud y latitud) y la fecha en la que se ha detectado el metal.

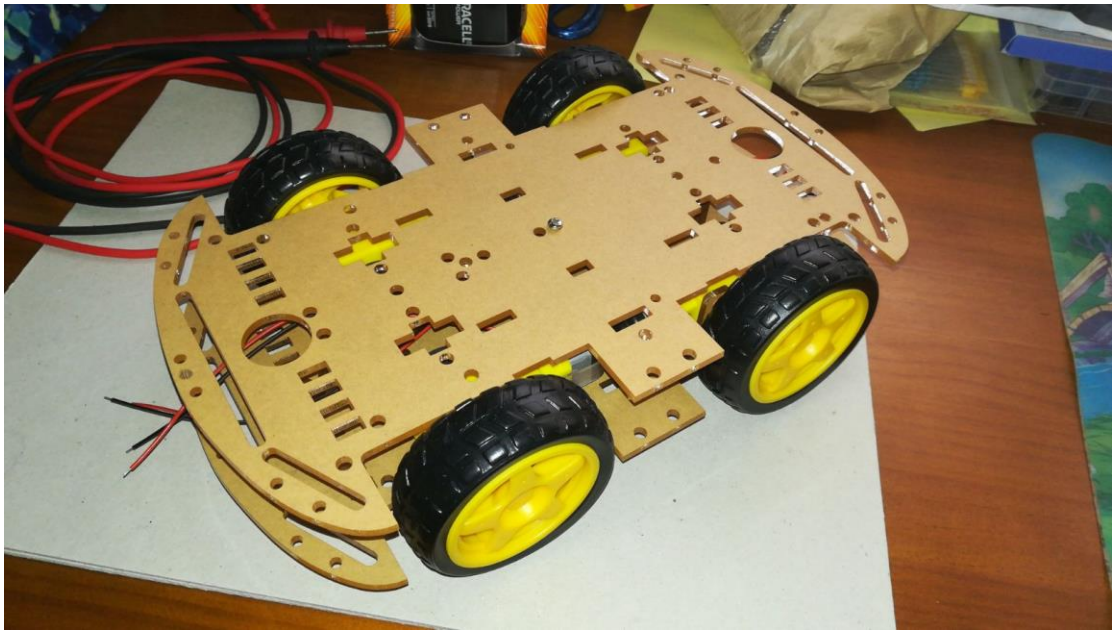
Este proyecto se nos ocurrió mirando en distintas páginas web en las que venían muchos proyectos. Vimos un detector de metales y nos pareció interesante la idea, pero decidimos hacerlo móvil, es por ello que lo hemos introducido en un coche, que era otra de las opciones que teníamos y que nos agradaba bastante.

2-Desarrollo del proyecto.

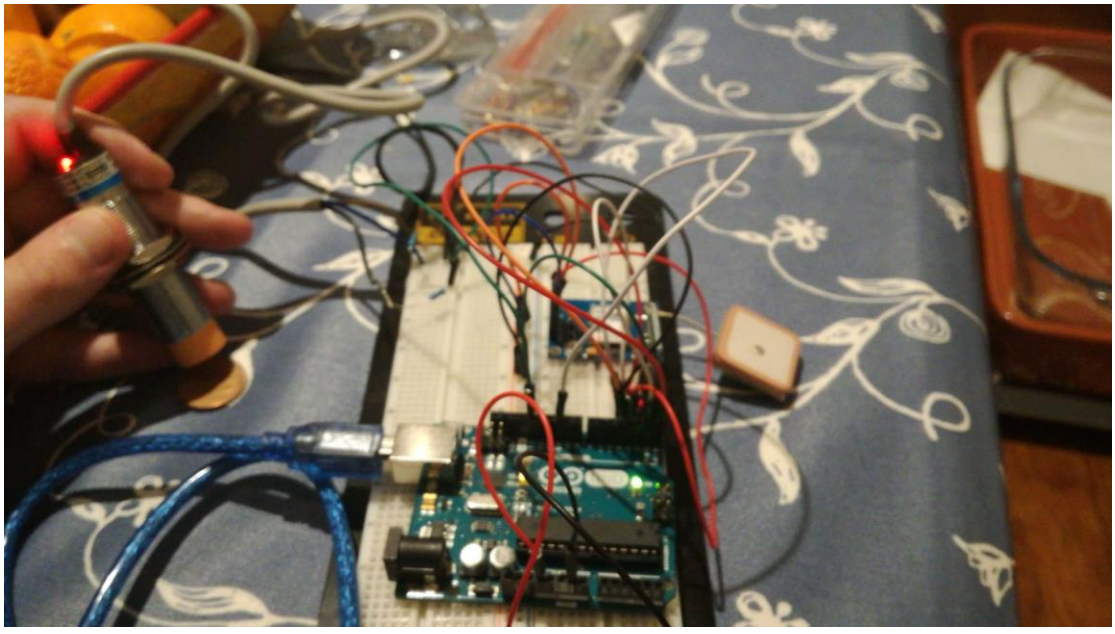
Una vez elegido el proyecto que queríamos hacer, pensamos qué necesitábamos con exactitud, pues nos habíamos hecho una idea general al principio. Para controlar el coche, mediante el módulo bluetooth en principio creamos una interfaz de java que finalmente no usamos por la simplicidad y comodidad de controlarlo con el móvil.

Además, montamos el chasis y antes de esto, soldamos cables a los motores para poder alimentarlos.





Probamos los componentes para comprobar que funcionaran correctamente y una vez comprobados, los juntamos para comprobar que el código necesario no se pisaba, ni daba fallos todo junto.



Para recibir los datos y leerlos, con el otro módulo bluetooth estuvimos pensando qué hacer, pues no sabíamos muy bien y finalmente nos decantamos por crear una aplicación para el móvil.

Para pasarle el voltaje necesario al módulo GPS hemos usado un regulador de tensión.

Posteriormente, colocamos todo en el chasis, lo cual nos costó bastante pues teníamos que ubicar muchas cosas en un espacio relativamente pequeño. Para ello, optamos por eliminar la protoboard y lo hemos conectado todo a la placa de Arduino UNO.

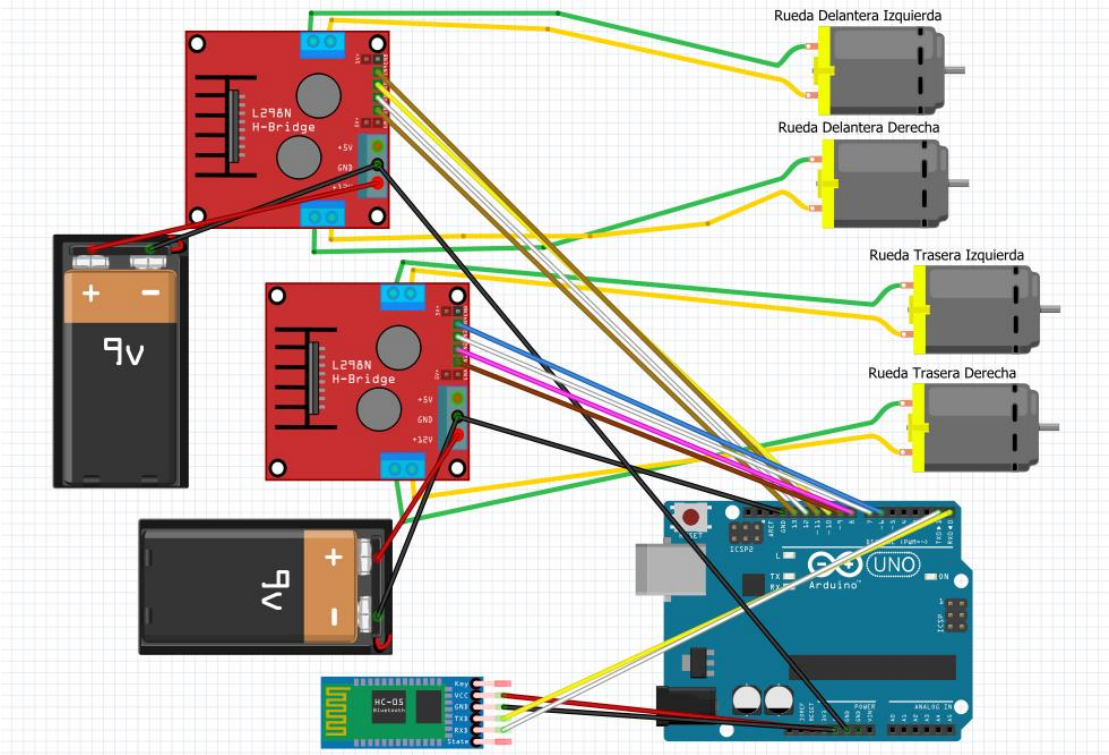
Una vez montado todo, hemos grabado el vídeo de prueba y hemos visto que el peso no frenaba completamente los motores, es decir, el coche seguía moviéndose a pesar del peso final del coche.

3-Presupuesto.

Componente	Coste
Arduino (x2)	En propiedad
Cables y resistencias	En propiedad
Regulador de tensión LM317	En propiedad
Pilas de botón de 9V (x2)	En propiedad
Pilas de petaca de 4,5V (x6)	En propiedad
Controlador de motores L298N (x2)	6'68€/unidad
Módulo bluetooth HC-05 (x2)	8'99€/unidad
Kit chasis del coche	13'99/unidad
Sensor inductivo LJ18 A3-8-Z/BX	7'99€/unidad
Módulo GPS NEO-6M	14'99€/unidad
Total	68'31€

4-Implementación.

4.1-CIRCUITO CONTROLADOR DEL COCHE



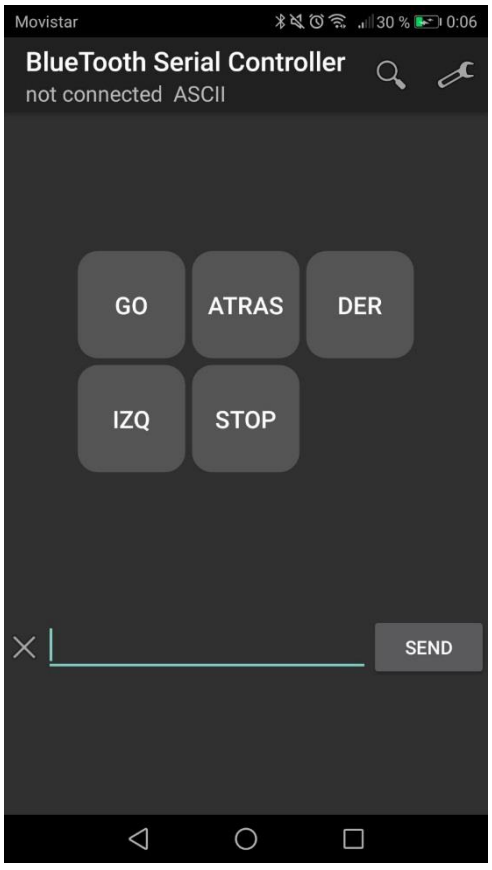
Para el circuito tendremos básicamente dos partes, por un lado, el módulo bluetooth el cual se conectará al teléfono móvil a través de una aplicación la cual posee unos botones que al pulsarlos mandaran una instrucción por el puerto serie. Dicha instrucción, llegará a nuestro Arduino y en función del botón pulsado llegará una instrucción u otra y activará los motores.

Conexiones Modulo Bluetooth a Arduino UNO	
Modulo Bluetooth	Placa Arduino Uno
Rx	Tx
Tx	Rx
Vcc	3.3V
Gnd	Gnd

Por otro lado, tenemos la parte de los motores, para controlar los cuatro motores tenemos dos controladores L298N, los cuales nos permiten activar unas ruedas u otras. Usaremos un controlador para las ruedas delanteras y otro para las traseras, estos controladores serán conectados a una alimentación de 9V cada uno. Además, cada uno posee 8 pines de control, 4 de entrada y otros 4 de salida, los de salida irán conectados a los motores y los de entrada a los pines Arduino que mandarán señales de HIGH o LOW para activar unos motores u otros, incluido su sentido de giro.

		OUT1	OUT2	OUT3	OUT4	IN1	IN2	IN3	IN4
Ruedas Delanteras	Derecha	Cable negro	Cable rojo			PIN 13	PIN 12		
	Izquierda			Cable rojo	Cable negro			PIN 10	PIN 11
Ruedas Traseras	Derecha	Cable negro	Cable rojo			PIN 9	PIN 8		
	Izquierda			Cable rojo	Cable negro			PIN 7	PIN 6

La imagen que aparece a continuación, muestra la aplicación empleada para controlar el coche y los motores, por ejemplo si pulsamos el botón “Go”, enviará por el puerto serie la instrucción “Avanzar” que llegara por bluetooth a nuestro Arduino.



```
/*
 * VAMOS A CONTROLAR LOS MOTORES MEDIANTE COMANDOS Y ADEMAS
 * UTILIZAREMOS UNA APLICACION ANDROID PARA QUE MANDE DICHOS
 * COMANDOS POR BLUETOOTH A NUESTRO SENSOR POR PUERTO SERIAL.
 */
int dd1=13;
int dd2=12;
int di1=11;
int di2=10;
int ti1=9;
int ti2=8;
int td1=7;
int td2=6;
char caracter;
String instruccion;

void setup() {

  Serial.begin(9600);
}

void loop() {
  while(Serial.available()>0){
    caracter = Serial.read();
    instruccion.concat(caracter);
    delay(10);
  }
```

```
if(instruccion.equals("avanzar")==true){
  digitalWrite (dd1, HIGH);
  digitalWrite (dd2, LOW);

  digitalWrite (di1, HIGH);
  digitalWrite (di2, LOW);

  digitalWrite (td1, HIGH);
  digitalWrite (td2, LOW);

  digitalWrite (ti1, HIGH);
  digitalWrite (ti2, LOW);

  Serial.println("Coche avanzando");
}
if(instruccion.equals("atras")==true){
  digitalWrite (dd1, LOW);
  digitalWrite (dd2, HIGH);

  digitalWrite (di1, LOW);
  digitalWrite (di2, HIGH);

  digitalWrite (td1, LOW);
  digitalWrite (td2, HIGH);

  digitalWrite (ti1, LOW);
  digitalWrite (ti2, HIGH);
```

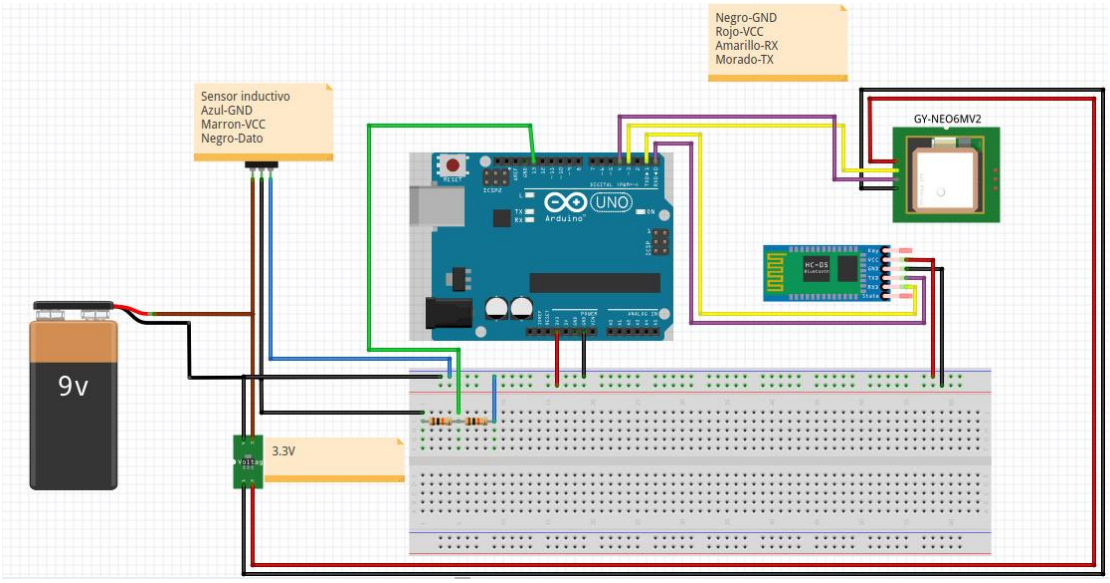
```
  Serial.println("Marcha atras");
}

if(instruccion.equals("derecha")==true){
  digitalWrite (dd1, HIGH);
  digitalWrite (dd2, LOW);

  digitalWrite (di1, LOW);
  digitalWrite (di2, LOW);

  digitalWrite (td1, HIGH);
  digitalWrite (td2, LOW);

  digitalWrite (ti1, LOW);
  digitalWrite (ti2, LOW);

  Serial.println("Girando a la derecha");
}
if(instruccion.equals("izquierda")==true){
  digitalWrite (dd1, LOW);
  digitalWrite (dd2, LOW);

  digitalWrite (di1, HIGH);
  digitalWrite (di2, LOW);

  digitalWrite (td1, LOW);
  digitalWrite (td2, LOW);
```

```
  digitalWrite (ti1, HIGH);
  digitalWrite (ti2, LOW);

  Serial.println("Girando a la izquierda");
 }

 if(instruccion.equals("parar")==true){
  digitalWrite (dd1, LOW);
  digitalWrite (dd2, LOW);

  digitalWrite (di1, LOW);
  digitalWrite (di2, LOW);

  digitalWrite (td1, LOW);
  digitalWrite (td2, LOW);

  digitalWrite (ti1, LOW);
  digitalWrite (ti2, LOW);

  Serial.println("Coche parado");
 }

 instruccion="";
}
```

En cuanto al código, lo que hacemos es leer la cadena de entrada por el puerto serie, cuando este tiene un dato disponible, dicha cadena de entrada se lee carácter a carácter

hasta que el serial deja de estar “available” que querrá decir que ha leído toda la cadena. Posteriormente, tras haber leído una cadena, entrará por una de las condiciones “if”, cuando entre en función de la instrucción hará una cosa u otra, por ejemplo, la condición “instruccion.equals("avanzar")==true”, hará que se active la polaridad de los motores de forma que avance hacia delante y todo ello hasta que pulsemos el botón de parar que lo detendrá. Otro ejemplo, sería la condición “instruccion.equals("derecha")==true”, esta activará solo las rudas del lado derecho, de forma que nos permita girar el coche hacia la derecha.

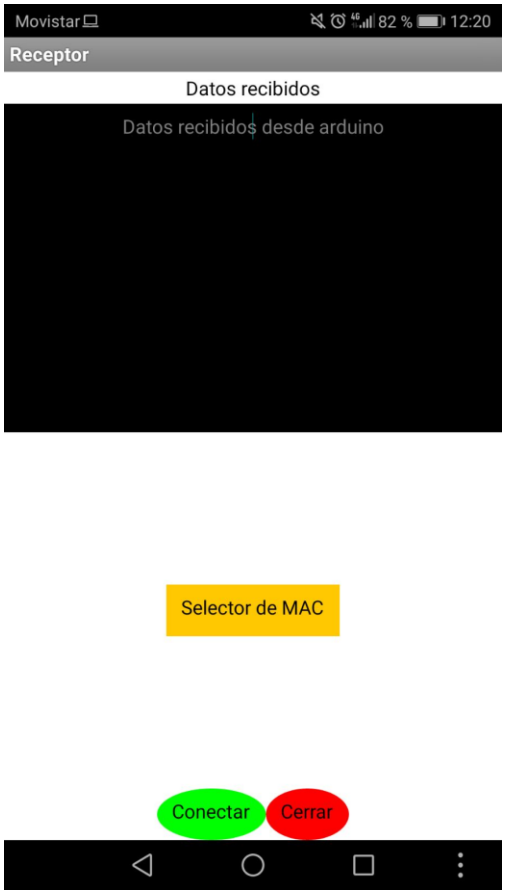
4.2-CIRCUITO DETECTOR DE METALES Y GPS



Para este circuito, tendremos un sensor inductivo que dará una señal baja cuando detecte un metal, y una señal alta cuando no detecte nada, un módulo GPS que se conectara con los satélites a través de la antena de cerámica, y un módulo bluetooth que enviará los datos del GPS al móvil cuando el sensor detecte un metal.

El sensor inductivo necesita una alimentación entre 6-36 voltios, por lo que hemos optado por una fuente de alimentación de 9V, y por falta de potencia del Arduino a la hora de alimentar el modulo bluetooth y el modulo GPS, hemos optado por conectar el modulo GPS en la alimentación de 9V con ayuda de un regulador de tensión variable fijado a 3.3V.

El modulo bluetooth se conecta al Arduino de la misma forma que el circuito anterior, el modulo GPS se conecta su Rx al pin 3 y su Tx al pin 4 del Arduino, y el sensor inductivo se conecta al pin 13 con ayuda de un divisor de tensión, ya que con la entrada de 9V, la salida del sensor superaba los 6V.



Hemos creado una pequeña aplicación para recibir los datos del Arduino, primero se seleccionaría la MAC del dispositivo bluetooth al que nos vamos a conectar, y a continuación, nos conectaríamos.

Una vez conectados, la aplicación del móvil enviará cada segundo un carácter de control para prepararse para recibir datos del Arduino, una vez que llegan los datos, la aplicación los muestra en el cuadrado negro, primero la latitud y longitud, con 2 decimales (para que no hubiera salto de línea, se puede alargar la tira para más precisión), seguidos de la fecha, hora, minuto y segundo en el que descubrió el metal.

Este dato se borra a los 5 segundos, para evitar superposición cuando se reciben muchos datos.

```
#include <SoftwareSerial.h>
#include <TinyGPS.h>
#include <string.h>

#define Sensor 13

boolean estado;
TinyGPS gps;
SoftwareSerial serialgps(4,3);

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

  pinMode(Sensor, INPUT);

  estado = false;
}

void loop() {
  if (!digitalRead(Sensor) && !estado) {
      delay(500);
      char comand = Serial.read();
      gps_data();
    }
  if (digitalRead(Sensor)) {
      estado = false;
    }
```

```
}

void gps_data() {
 smartdelay(1000);

 float flat, flon;
 unsigned long age;
 int year;
 byte month, day, hour, minute, second, hundredths;

 gps.f_get_position(&flat, &flon, &age);

 Serial.print("Lat/Lon:");
 if (flat == TinyGPS::GPS_INVALID_F_ANGLE) {
  Serial.print("Invalid");
 } else {
  Serial.print(flat,2);
  smartdelay(0);
 }
 Serial.print("/");

 if (flon == TinyGPS::GPS_INVALID_F_ANGLE) {
  Serial.print("Invalid");
 } else {
  Serial.print(flon,2);
  smartdelay(0);
 }
```

```
  Serial.print("   TIME: ");


  gps.crack_datetime(&year, &month, &day, &hour, &minute, &second, &hundredths,
&age);


  if (age == TinyGPS::GPS_INVALID_AGE) {
    Serial.println("UNKNOWN");
  } else {
    char sz[32];
    sprintf(sz, "%02d/%02d/%02d %02d:%02d:%02d ", month, day, year, hour, minute,
second);
    Serial.println(sz);
    smartdelay(0);
  }


}


static void smartdelay(unsigned long ms)
{
  unsigned long start = millis();
  do
  {
    while (serialgps.available())
      gps.encode(serialgps.read());
  } while (millis() - start < ms);
}
```

El código funciona de la siguiente manera:

Primero incluimos las librerías de SoftwareSerial, para crear dos puertos seriales extra vía software, en los pines 3 y 4, y TinyGps, una librería para recoger los datos del módulo GPS.

A continuación, en el void setup, iniciamos el puerto serial del bluetooth, y el puerto serial para el bluetooth, utilizamos una variable booleana estado para evitar que mande varias veces un dato, y marcamos el pin 13 como entrada digital del sensor.

El código del void loop es bastante sencillo, si detecta un metal (entrada baja), esperara al carácter de control, y a continuación enviará los datos del módulo GPS, y cambiaremos la variable estado para evitar que vuelva a entrar en el bucle hasta que deje de detectar el metal actual.

El subprograma que se dedica al envío de datos del GPS, primero hace un pequeño delay de 1 segundo para poder sintetizar los datos recibidos por el satélite, y a continuación llama a un método de la librería tinyGps para coger latitud y longitud, que a continuación enviará (si hay algún problema, enviará "Invalid") con 2 decimales, aunque se podría aumentar para más precisión, y por último, gracias a otro método de la librería, cogería la fecha actual y la hora, minuto y segundo del descubrimiento (UNKNOWN en caso de algún error).

El subprograma smartdelay se ocupa de hacer una mini pausa para evitar que se corrompan los datos (siguen la idea de un subprograma idéntico del último ejemplo de la librería).

5-Problemas encontrados y soluciones.

Hemos visto que el módulo bluetooth adquirido no es compatible con iOS, por lo que una posible mejora del detector de metales sería usar un módulo bluetooth que sea compatible con dicho sistema operativo.

Cuando probamos el módulo GPS por primera vez, al no estar soldado de vez en cuando tenía problemas de contacto, sumado a la dificultad que tiene la antena de encontrar señal en interiores, era bastante complicado conseguir que mostrase alguna posición por pantalla, lo solucionamos soldando 4 pines al módulo, y haciendo las pruebas donde la aplicación GPS test nos dijese que hubiera buena señal.

El módulo bluetooth, por alguna razón que desconocemos, enviaba datos al Arduino mientras intentábamos cargar código a esta, se solucionaba simplemente quitando el pin emisor del bluetooth, desconocemos la causa real del problema, pero el módulo funcionaba perfectamente después de cargar el código.

El Arduino no es capaz de alimentar el módulo GPS y el módulo bluetooth al mismo tiempo, por lo que tuvimos que optar por un regulador de tensión que teníamos a mano, y alimentar el módulo bluetooth de forma externa al Arduino.

Al probar nuestro código original, no imprimía los datos del GPS por pantalla, lo solucionamos añadiendo un subprograma smartdelay, que encontramos explorando en los ejemplos de la librería tinyGps, concretamente el último ejemplo, en el que lo usaban cada vez que hacían una llamada a los subprogramas de la librería, y al inicio de 1 segundo, al utilizarla de la misma forma conseguimos que imprimiera los datos por pantalla.

A causa de un fallo en las conexiones, metimos de forma inversa 9V al módulo bluetooth quemando el componente, y Amazon nos podía traer otro módulo, pero 1 día después de la entrega, y probando de nuevo los componentes nuetro módulo bluetooth del GPS dejó de funcionar (desconocemos el motivo), por lo que para poder montar el coche y poder pasearlo por exteriores, pedimos el módulo GPS y un módulo bluetooth a un compañero, el cual ya lo uso previamente.

Por alguna razón que desconocemos, de un día para otro, las señales de los controladores a los motores se invierten, provocando en 2 ocasiones que el coche no girase, y obligándonos a cambiar las entradas del Arduino.

6-Posibles mejoras.

El empleo de una única aplicación para enviar y recibir datos (no tuvimos tiempo de realizarla por el percance del módulo GPS).

Hacer que el coche se detenga si pierde la señal bluetooth del emisor.

Añadir un chasis para evitar que los cables estén al aire libre.

Añadirle un sensor ultrasónico para evitar que se choque.