UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

mim máster en ingeniería mecatrónica

ADQUISICIÓN Y PROCESADO DE INFORMACIÓN DE POSICIONAMIENTO GPS MEDIANTE DISPOSITIVO INALÁMBRICO BASADO EN ARDUINO

MEMORIA PRESENTADA POR:

Jorge Alambiaga Pascual

Máster Universitario en Ingeniería Mecatrónica

DIRECTOR:

Vicente Fermín Casanova Calvo

Septiembre 2017

ÍNDICE

1. OBJETO DEL PROYECTO

El objetivo del presente proyecto final de Máster es el desarrollo de una aplicación de bajo coste con la ayuda de un kit módulo receptor GPS de arduino, un módulo bluetooth y un receptor ordenador o smartphone, para más tarde obtener las diferentes tramas de datos de forma inalámbrica y después analizar los datos obtenidos para mostrar una localización o ruta realizada en un tiempo determinado en una interfaz gráfica en Matlab

El desarrollo de este proyecto tiene la finalidad de entender como funcionan los gps, que son y todas las aplicaciones que se puede realizar con ellos al obtener las diferentes tramas de datos que recibe un módulo gps de los satélites ubicados en la órbita terrestre.

La realización del presente proyecto tiene como finalidad la de ser presentado en la Escuela Superior de Ingeniería del Diseño de la Universidad Politécnica de Valencia como Proyecto Final de Máster y cumplir con los requerimientos necesarios para la obtención del Título de Máster en Ingeniería Mecátronica.

2.JUSTIFICACIÓN

La realización de este TFM viene motivada por la necesidad de conocer el funcionamiento de los gps y conocer las aplicación que puede suponer en el desarrollo de nuevas tecnologías como vehículos guiados de forma autónoma o simplemente el posicionamiento en el que se encuentren respecto al globo terrestre de personas, edificios , barcos, drones y el trazado de diferentes rutas realizadas o sin realizar para obtener parámetros como latitud, longitud, velocidad o altitud sobre el nivel del mar

3.ANTECEDENTES

3.1 Composición y funcionamiento

El sistema de posición global más conocido como GPS utiliza un Sistema Global de Navegación por Satélite (GNSS) que consiste en una constelación de satélites que transmite señales de radio utilizadas para el posicionamiento y localización en cualquier parte del globo terrestre, ya sea en tierra, mar o aire.

Estos aparatos permiten determinar las coordenadas geográficas, altitud, velocidad respecto del suelo, de un punto dado como resultado de la recepción de señales provenientes de las constelaciones de satélites artificiales que rodean el globo terrestre.

Fue desarrollado por el Departamento de Defensa de Estados Unidos. El aparato receptor G.P.S. es lo que se suele conocer con el simple nombre de G.P.S. que permite la navegación por cualquier lugar de la tierra de una forma muy sencilla, con un coste gratuito y con gran precisión, por lo que su uso se ha popularizado rápidamente en todos los ámbitos, desde la geodesia, la ingeniería, la navegación marítima, el excursionismo o el alpinismo.

Este sistema G.P.S. emplea 24 satélites que orbitan sobre el globo terrestre a 20.200 kilómetros de altitud, y que forman la constelación Navstar. Los satélites se ubican sobre seis órbitas prácticamente circulares (con una excentricidad de 0’03).

En cada órbita se sitúan cuatro satélites con una separación de 90º entre cada uno de ellos, para tener al menos 4 satélites a la vista para conocer nuestra posición.

Ilustración 1. Constelación de satelites

Con un solo satélite no será posible obtener la posición, se necesitará al menos tres satélites de la constelación, para poder realizar una triangulación de señales y conocer con mayor exactitud la posición. El funcionamiento es el siguiente:

Cada satélite indica que el receptor se encuentra en un punto en la superficie de la esfera, con centro en el propio satélite y de radio la distancia total hasta el receptor.

Obteniendo información de al menos dos satélites más, queda determinada una circunferencia que resulta cuando se intersecan las esferas en algún punto de la cual se encuentra el receptor.

Triangulación GPS

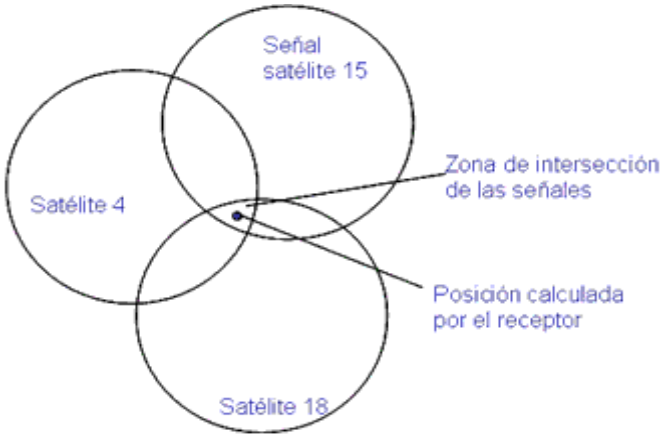


Ilustración 2. Triangulación de posición

También se puede obtener la altitud del punto. Para ello es necesario disponer de un satélite más. Como mínimo cuatro satélites para la navegación tridimensional (que incluye la altitud) y sólo tres satélites para la navegación bidimensional (sin altitud) sobre la superficie terrestre.

El receptor GPS funciona midiendo su distancia de los satélites, y usa esa información para calcular su posición. Esta distancia se mide calculando el tiempo en que la señal viaja y tarda en llegar al receptor. Conocido este tiempo y basándose en el hecho de que la señal viaja a la velocidad de la luz (dependiendo las correcciones que se aplican debido a los gases de la atmósfera), se puede calcular la distancia entre el receptor y el satélite. también es interesante comprobar que el tiempo necesario para que una señal llegue de un satélite al receptor G.P.S. es sumamente pequeño pero imprescindible. Siendo la velocidad de la luz c=300.000 km/s, este tiempo es del orden de:

t = 20.200 km / 300.000 km/s = 0’067 s = 67 ms

Ilustración 3. Receptor gps

La comunicación entre satélites y receptor se realiza comúnmente mediante el envio de paquetes con sentencias conocidos como comandos NMEA (National Marine Electronic Asociation). Asociación que se encarga de definir un protocolo estándar de comunicación de datos, la última versión NMEA-0183 del año 2001.

Todos los datos son transmitidos en sentencias con caracteres ASCII, en cada sentencia comienza con el símbolo ‘$’ y termina con ‘CR o LF’. Los siguientes 2 caracteres después de ‘$’ son los que identifican al equipo, ‘GP’ (por ejemplo: se utiliza para identificar datos GPS), los tres siguientes caracteres que le siguen son el identificador

del tipo de sentencia que se envía. Los tipos de sentencias NMEA que existen corresponden a envio, origen del equipo y consulta.

Los datos están delimitados por comas dentro de la sentencia. Dentro de estas sentencias de abajo se encuentran los diferentes datos que se utilizan para obtener la localización y demás datos.

```
$GPGGA,125652.000,3941.9558,N,00031.9250,W,1,09,0.9,282.5,M,51.9,M,,0000*4B
$GPGSA,A,3,08,27,10,11,01,22,32,18,28,,,,1.6,0.9,1.3*31
$GPRMC,125652.000,A,3941.9558,N,00031.9250,W,0.00,142.43,280817,,,A*7D
$GPGGA,125653.000,3941.9558,N,00031.9250,W,1,09,0.9,282.5,M,51.9,M,,0000*4A
$GPGSA,A,3,08,27,10,11,01,22,32,18,28,,,,1.6,0.9,1.3*31
$GPRMC,125653.000,A,3941.9558,N,00031.9250,W,0.00,142.43,280817,,,A*7C
$GPGGA,125654.000,3941.9558,N,00031.9251,W,1,09,0.9,282.5,M,51.9,M,,0000*4C
$GPGSA,A,3,08,27,10,11,01,22,32,18,28,,,,1.6,0.9,1.3*31
$GPRMC,125654.000,A,3941.9558,N,00031.9251,W,0.00,142.43,280817,,,A*7A
$GPGGA,125655.000,3941.9558,N,00031.9251,W,1,09,0.9,282.5,M,51.9,M,,0000*4D
$GPGSA,A,3,08,27,10,11,01,22,32,18,28,,,,1.6,0.9,1.3*31
$GPGSV,3,1,12,08,76,337,38,27,60,110,38,10,43,052,40,11,38,289,36*7D
$GPGSV,3,2,12,01,29,268,37,22,24,208,38,32,20,110,32,18,15,045,26*74
$GPGSV,3,3,12,28,08,325,31,04,00,000,30,16,22,178,26,39,39,141,*70
$GPRMC,125655.000,A,3941.9558,N,00031.9251,W,0.00,142.43,280817,,,A*7B
$GPGGA,125656.000,3941.9558,N,00031.9251,W,1,09,0.9,282.5,M,51.9,M,,0000*4E
$GPGSA,A,3,08,27,10,11,01,22,32,18,28,,,,1.6,0.9,1.3*31
$GPRMC,125656.000,A,3941.9558,N,00031.9251,W,0.00,142.43,280817,,,A*78
```

Ilustración 4. Diferentes tramas NMEA

Todo esto a una velocidad de 4.800 baudios para una vez transmitidas estas tramas al GPS ya se encarga un software de mostrar los datos ordenados de posición y otros.

Existen multitud de tramas dependiendo del objeto que vaya a recibirlas las más comunes e importantes se van a nombrar, explicar y localizar la posición que ocupa en la trama separada por comas a continuación:

3.2 TRAMAS NMEA

GGA

GGA - datos de corrección esenciales que proporcionan datos de localización y exactitud en 3D.

$ GPGGA, 123519,4807.038, N, 01131.000, E, 1,08,0,9,545.4, M, 46,9, M ^ {47}

Dónde:

Sistema de posicionamiento global GGA Fix Data
123519 Ficha hecha a las 12:35:19 UTC
4807.038, N Latitud 48 grados 07.038 'N
01131.000, E Longitud 11 grados 31.000 'E
1 Calidad de la fijación: 0 = inválido
1 = Fijación GPS (SPS)
2 = Ajuste DGPS

3 = corrección de PPS
4 = Cinemática en tiempo real
5 = Floating RTK
6 = estimado (cuenta muerta) (característica 2.3)
7 = Modo de entrada manual
8 = Modo de simulación
08 Número de satélites que están siendo rastreados
0.9 Dilución horizontal de la posición
545.4, M Altitud, Metros, por encima del nivel medio del mar
46.9, M Altura del geoide (nivel medio del mar) por encima de WGS84 Elipsoide
(Campo vacío) en segundos desde la última actualización DGPS
(Campo vacío) Número de identificación de la estación DGPS
* 47 los datos de la suma de comprobación, comienza siempre con *

GSA

GSA - GPS DOP y satélites activos. Esta sentencia proporciona detalles sobre la naturaleza de la corrección. Incluye los números de los satélites que se utilizan en la solución actual y el DOP. DOP (dilución de precisión) es una indicación del efecto de la geometría del satélite sobre la exactitud de la corrección. Es un número sin unidad donde más pequeño es mejor. Para los arreglos en 3D usando 4 satélites, se consideraría que un 1,0 sería un número perfecto, sin embargo para soluciones sobredeterminadas es posible ver números por debajo de 1.0.

$ GPGSA, A, 3,04,05,, 09,12 ,,, 24 ,,,,, 2,5,1,3,2,1 * 39

Dónde:
Estado del satélite GSA
A Selección automática de fijación 2D o 3D (M = manual)
3 fijar 3D - los valores incluyen: 1 = no fijar
2 = fijación 2D
3 = Solución 3D
04,05 ... PRN de los satélites utilizados para la fijación (espacio para 12)
2,5 PDOP (dilución de precisión)
1.3 Dilución horizontal de precisión (HDOP)
2.1 Dilución vertical de precisión (VDOP)
* 39 los datos de la suma de comprobación, comienza siempre con *

GSV

GSV - Satélites a la vista, muestra datos sobre los satélites que la unidad podría encontrar en función de su máscara de visualización y datos de almanaque. También muestra la capacidad actual para rastrear estos datos. Tenga en cuenta que una oración GSV sólo puede proporcionar datos de hasta 4 satélites y, por lo tanto, puede ser necesario que haya 3 oraciones para la información completa. Es razonable que la sentencia GSV contenga más satélites que GGA podría indicar ya que GSV puede incluir satélites que no se usan como parte de la solución. No es un requisito que las oraciones GSV aparecen todas en secuencia. Para evitar sobrecargar el ancho de banda

de datos algunos receptores pueden colocar las diversas oraciones en muestras totalmente diferentes ya que cada oración identifica cuál es.

$ GPGSV, 2,1,08,01,40,083,46,02,17,308,41,12,07,344,39,14,22,228,45 * 75

Dónde:

GSV Satélites en vista

2 Número de frases para los datos completos

1 oración 1 de 2

08 Número de satélites a la vista

01 Número de satélite PRN

40 Elevación, grados

083 Azimut, grados

46 SNR - más alto es mejor

Hasta 4 satélites por oración

* 75 los datos de la suma de comprobación, comienza siempre con *

RMC

RMC - NMEA tiene su propia versión de los datos esenciales de gps pvt (posición, velocidad, tiempo). Se llama RMC, el mínimo recomendado, que se verá similar a:

$ GPRMC, 123519, A, 4807.038, N, 01131.000, E, 022.4.084.4.230394.003.1, W * 6A

Dónde:

RMC Recomendó la sentencia mínima C

123519 Ficha hecha a las 12:35:19 UTC

A Estado A = activo o V = vacío.

4807.038, N Latitud 48 grados 07.038 'N

01131.000, E Longitud 11 grados 31.000 'E

022.4 Velocidad sobre el suelo en nudos

084.4 Ángulo de la pista en grados Verdadero

230394 Fecha - 23 de marzo de 1994

003.1, W Variación magnética

* 6A Los datos de la suma de comprobación, empiezan siempre con *

GLL

GLL - La latitud y la longitud geográficas son una reserva de datos de Loran y algunas unidades viejas no pueden enviar la información activa del tiempo y de los datos si están emulando datos de Loran. Si un GPS está emulando datos de Loran pueden usar el prefijo LC Loran en lugar de GP.

$ GPGLL, 4916,45, N, 12311,12, W, 225444, A, * 1D

Dónde:

GLL Posición geográfica, latitud y longitud

4916.46, N Latitud 49 grados. 16,45 min. norte

12311.12, W Longitud 123 grados. 11,12 min. Oeste

225444 Arreglo tomado a las 22:54:44 UTC

Un Data Active o V (void)

* Datos de suma de comprobación iD

VTG

VTG - Velocity hecho bueno. El receptor de gps puede usar el prefijo LC en lugar de GP si está emulando la salida de Loran.

$ GPVTG, 054,7, T, 034,4, M, 005,5, N, 010,2, K * 48

dónde:

Velocidad VTG Track buena y baja

054.7, T Pista verdadera buena (grados)

034.4, M Pista magnética hecha bien

005.5, N Velocidad de tierra, nudos

010.2, K Velocidad del suelo, Kilómetros por hora

* 48 Checksum

4. MATERIALES UTILIZADOS

Arduino uno:

Es una plataforma de hardware libre, está permitido usar el diseño y distribución de forma libre sin requerir licencia.

Posee un lenguaje de programación basado en C/C++ con lo que esta soportado por muchos sistemas operativos.

Este es el modelo de arduino que más se utiliza por los usuarios cuando empiezan a utilizar arduino, la placa se compone de un microcontrolador Atmel, con unas

dimensiones de 68.6x53.4 mm. Existen dos tipos de formatos DIP y SMD en este caso se utilizará el formato DIP.

Ilustración 5. Microcontrolador arduino uno formato DIP

Microcontrolador	ATmega328P
Voltaje de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (limite)	6-20V
Pines digitales E/S	14 (de los cuales 15 proporcionan salidas PWM)
Pines digitales PWM E/S	6
Pines analógicos E/S	6
Corriente DC para pines E/S	20 mA
Corriente DC para pines de 3.3V	50 mA
Memoria Flash	32 KB de los cuales 8KB usados por bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Velocidad del reloj	16 MHz
LED_BUILTIN	13
Largo	68.6 mm
Ancho	53.4 mm
Peso	25 g

Ilustración 6. Características Arduino uno

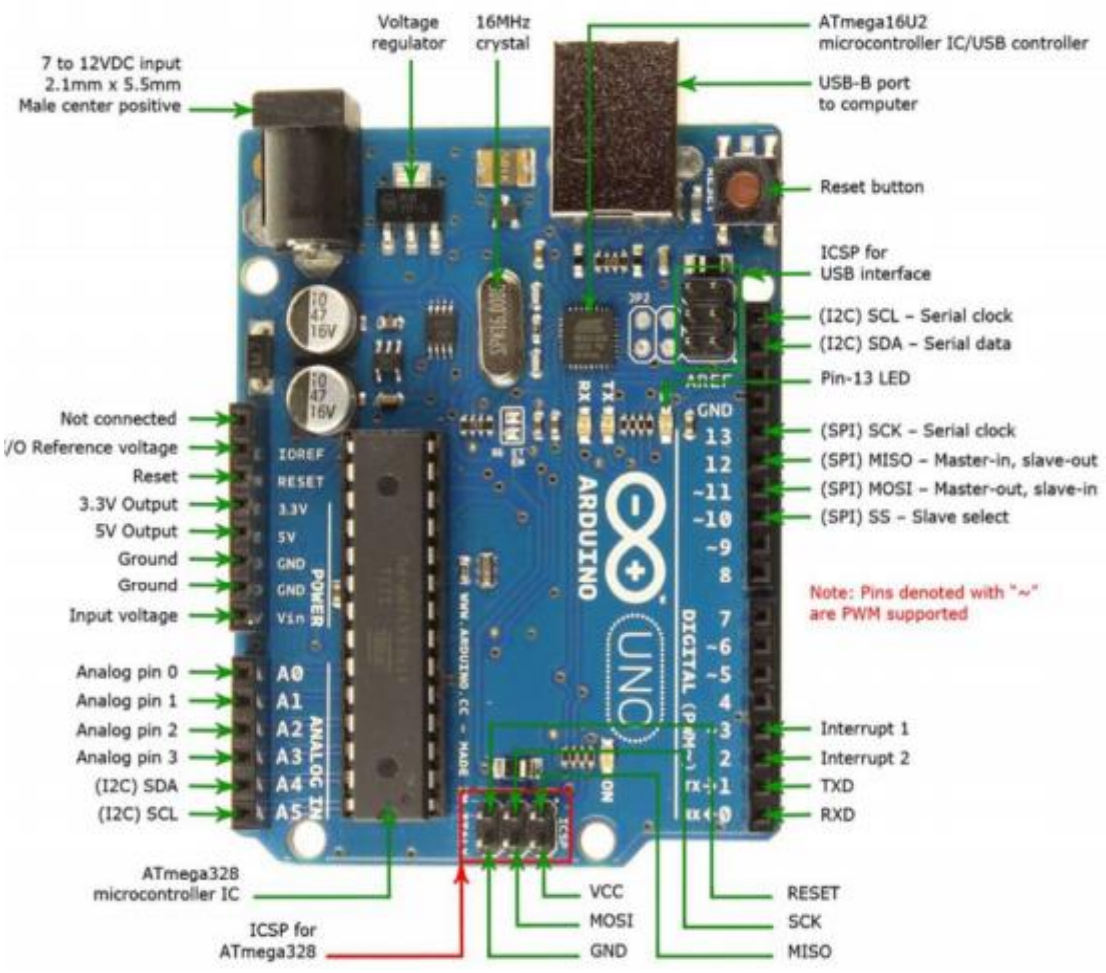


Ilustración 7. Distribución elementos de arduino

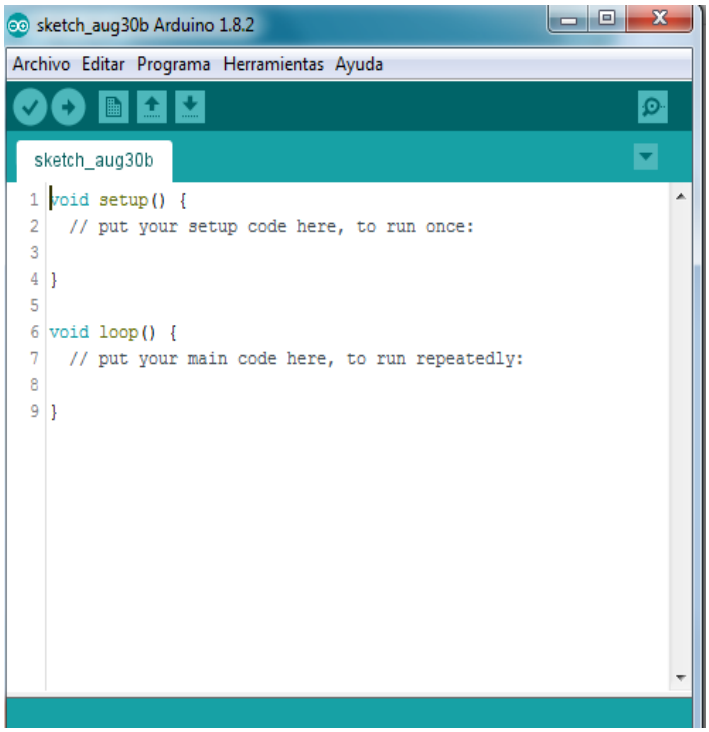


Ilustración 8. Entorno programación arduino

Matlab

MATLAB es un entorno de cálculo técnico de altas prestaciones para cálculo numérico y visualización. Integra:

Análisis numérico

Cálculo matricial

Procesamiento de señales

Gráficos

Con un lenguaje de programación propio (lenguaje M).parecido al C/C++

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware .

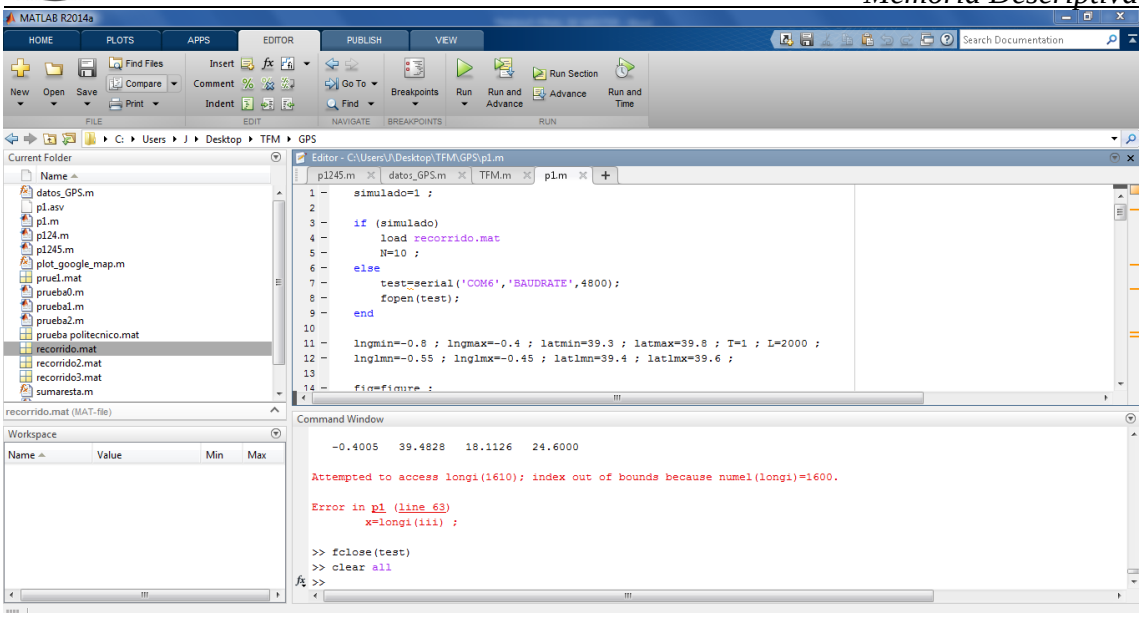


Ilustración 9. Entorno programación Matlab

5. DESARROLLO DEL PROYECTO

1º PASO

El primer paso para comenzar con el desarrollo del proyecto fue tener en mano todos los componentes necesarios que forman el kit GPS Sparkfun, el receptor GPS con el escudo y la placa del microcontrolador Arduino uno.

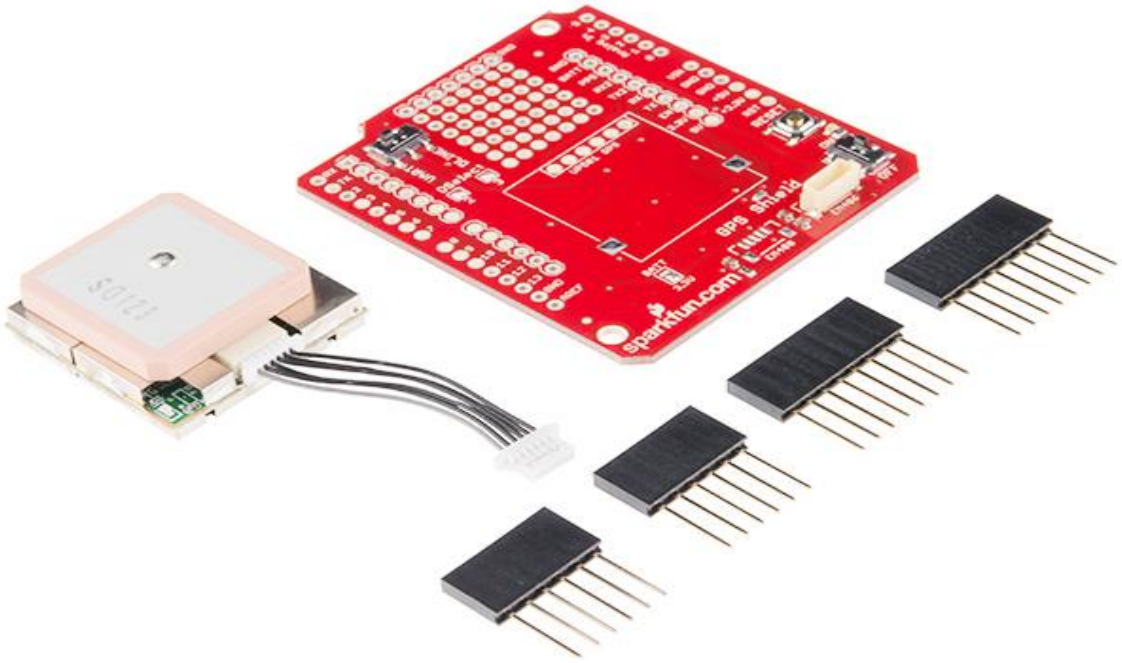


Ilustración 10. Sparkfun GPS Starter Kit

Una vez conexionados y montados entre ellos quedaría conforme a la siguiente imagen:

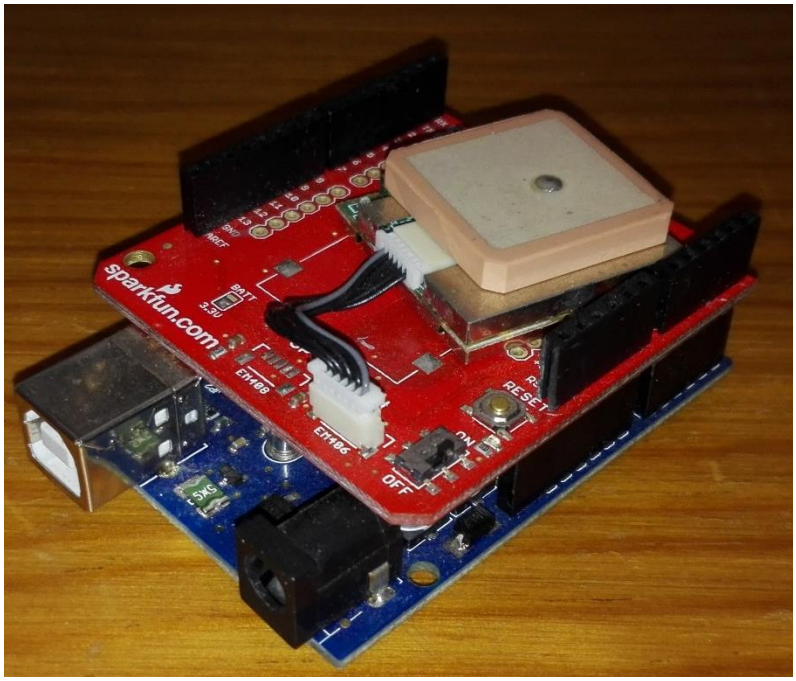


Ilustración 11. Arduino con escudo gps

Ahora se conectaría mediante un cable usb al ordenador y con el software arduino ide 1.8.2 se introduciría un código de programación utilizando la librería TinyGPS para comprobar que funciona correctamente y que recibe información respecto de los satélites adecuadamente por el puerto serie de arduino.

CODIGO_gps

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
TinyGPSPlus gps;
SoftwareSerial ss(2,3);

void setup() {
   Serial.begin(4800);
 ss.begin(4800);
}
void loop() {

while (ss.available() > 0)
  gps.encode(ss.read());
if (gps.altitude.isUpdated())
{
Serial.print("Altitud= ");Serial.println(gps.altitude.meters());
Serial.print("Fecha= "); Serial.println(gps.date.value());
Serial.print("LATITUD= "); Serial.print(gps.location.lat(), 6);
Serial.print("\n");
Serial.print("LONGITUD= "); Serial.println(gps.location.lng(), 6);
Serial.print("Satélites disponibles= ");Serial.println(gps.satellites.value());
Serial.print("Velocidad= ");Serial.println(gps.speed.kmph());
Serial.print("\n");
delay(2000);
}
}
```

Ilustración 12. Código arduino con Tinygps

A continuación se muestran los datos que recibe el puerto serie de arduino decodificados por la librería y el código anterior ,muestreados cada 2 segundos.

```
Altitud= 276.50
Fecha= 290817
LATITUD= 39.699298
LONGITUD= -0.532068
Satélites disponibles= 8
Velocidad= 0.00

Altitud= 276.60
Fecha= 290817
LATITUD= 39.699298
LONGITUD= -0.532068
Satélites disponibles= 8
Velocidad= 0.00

Altitud= 276.60
Fecha= 290817
LATITUD= 39.699298
LONGITUD= -0.532068
Satélites disponibles= 8
Velocidad= 0.00
```

Ilustración 13. Datos puerto serie arduino

2º PASO

El segundo paso una vez comprobado que todos los datos recibidos en el puerto serie arduino eran correctos se tomó la decisión de que la transmisión de datos sería mejor realizarla de forma inalámbrica para no tener el inconveniente de ir siempre conectado mediante un cable y por ello escogimos un módulo bluetooth HC-05 con el que teníamos la posibilidad de transmitir los datos de forma inalámbrica pero también había que dotar al microcontrolador arduino de una batería al no estar conectado mediante usb la alimentación eléctrica se perdía.

Arduino al poseer un conector para alimentación de forma externa por medio de 9V de tensión, no fue un inconveniente ya que con una pila de 9V pudimos solventar el inconveniente.

Después para que la transmisión de datos se realizara de forma inalámbrica, se tenía que configurar el módulo bluetooth con los parámetros de: nombre de identificación, contraseña de enlace, velocidad de transmisión de datos y algunos otros.

Para poder realizar esto se debe de poner el módulo HC-05 en modo AT, que significa un modo para mandarle información mediante comandos para que se pueda configurar. En este modulo para entrar a ese modo posee un botón en el que hay que mantenerlo apretado antes de alimentarlo con electricidad y se sabe si ha entrado o no por la frecuencia de ilumnación de un led que posee.

Ilustración 14. Módulo bluetooth HC-05

AT : comprobar la conexión.

AT+NAME : ver el nombre por defecto.

AT+ADDR : ver la dirección por defecto.

AT+VERSION : ver la versión.

AT+UART : ver la velocidad de transmisión.

AT+ROLE: ver el rol del módulo bluetooth (1 = master, 0 = esclavo).

AT+RESET : reset y salir del modo AT.

AT+ORGL : restaurar a los configuración de fábrica.

AT+PSWD: ver la contraseña por defecto.

Hay muchos más pero estos son los más usados, solo se cambiaron unos pocos parámetros que se describen a continuación, los demás conforme venían configurados por defecto de fábrica servían.

AT+NAME=HC-05	nombre que se mostrará para la identificación del bluetooth
AT+PIN=1234	contraseña de enlace entre dispositivos
AT+BAUD=3	configurar velocidad de transmisión de datos a 4800bps

configbt05

```
#include <SoftwareSerial.h>
SoftwareSerial BT1(10, 11); // RX | TX
void setup(){
  pinMode(9,OUTPUT);
  digitalWrite(9,HIGH);
   Serial.begin(4800);
   Serial.println("Enter AT commands:");
   BT1.begin(38400);

   Serial.print("Inicio comandos AT:");
}
void loop()
   {  if (BT1.available())
            Serial.write(BT1.read());
      if (Serial.available())
            BT1.write(Serial.read());
   }
```

Ilustración 15. Código configuración HC-05

Para poder mandar los comandos AT es necesario el código de arriba para poder introducir los comandos AT desde el puerto serie de arduino para poder comunicarnos con el bluetooth.

Una vez configurado correctamente se probó la conexión entre el bluetooth HC-05 y el bluetooth del ordenador se enlazaron con la contraseña y luego comprobar que se enviaban correctamente los datos que recibía el gps se transmitían por el bluetooth HC-05 hacia un receptor bluetooth integrado en el ordenador por ejemplo.

Para esto escribimos un nuevo código en arduino para que los datos que se reciban del gps los transmita directamente por bluetooth.

dato_gps §

```
#include <SoftwareSerial.h>
SoftwareSerial gps(2,3);
SoftwareSerial bt(4,5);
char dato=' ';

void setup()
{
 Serial.begin(4800);
 gps.begin(4800);
 //bt.begin(4800);
}


void loop()
{
  if(gps.available())
  {
    dato=gps.read();
    Serial.write(dato);
  }
}
```

Ilustración 16. Código transmisión datos gps a bluetooth

Debido a que no se pueden transmitir bien los datos cuando se crea más de un puerto digital con la librería SoftwareSerial, se cambian los pines del bluetooth a 0 y 1 que es el puerto físico que tiene el arduino uno (Rx y Tx).

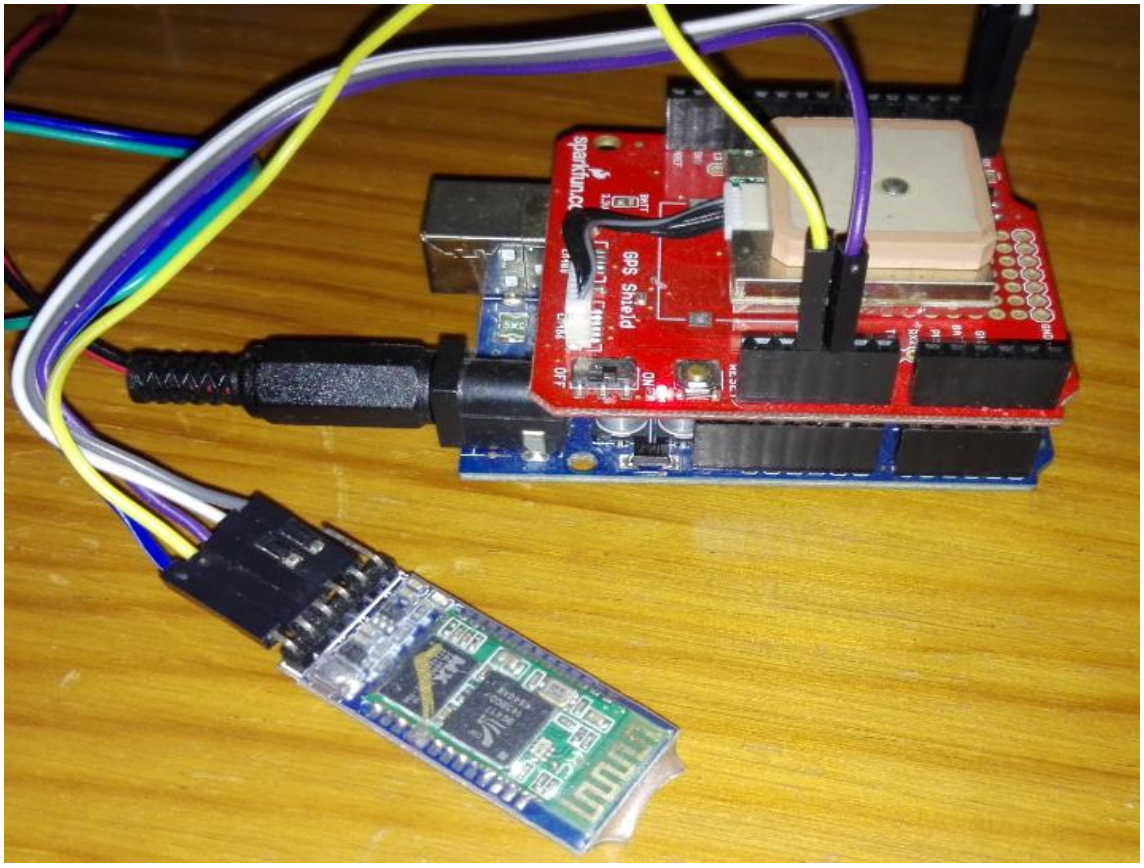


Ilustración 17. Conexión bluetooth con escudo

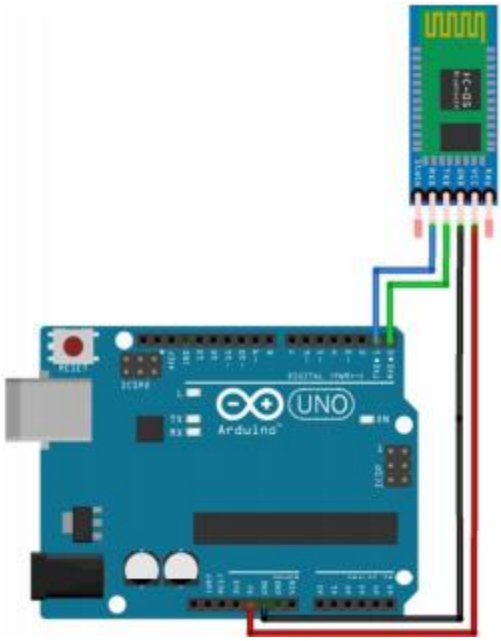


Ilustración 18. Esquema conexión bluetooth

3º PASO

El tercer paso comienza en abrir el programa Matlab y mediante la interfaz bluetooth que posee buscar el bluetooth HC-05 y crear un objeto para poder recibir datos y leerlos en el propio programa. Con los códigos siguientes se realizan estas funciones:

instrhwinfo('Bluetooth');

instrhwinfo('Bluetooth', RemoteName);

Aparece el HC-05 y se escribe el nombre del bluetooth y el canal en que transmite para crear un objeto bluetooth.

b = Bluetooth('HC-05', 1);

Se abre el objeto bluetooth con el comando siguiente para poder conectarse con él.

fopen(b)

y más tarde se escanea con

fscanf(b);

para leer los datos de texto que manda el HC-05.

y aparecen las tramas NMEA que manda el gps por bluetooth en el command window de Matlab.

```
$GPGGA,180857.000,3941.9564,N,00031.9254,W,1,08,1.2,266.9,M,51.9,M,,0000*49



ans =

$GPGSA,A,3,23,09,03,06,19,17,22,01,,,,,2.4,1.2,2.0*37



ans =

$GPGSV,3,1,12,23,76,058,30,09,70,220,26,03,45,068,35,06,44,311,30*70



ans =

$GPGSV,3,2,12,19,39,258,31,17,33,232,37,22,25,083,21,01,21,139,31*71
```

```
$GPGSA,A,3,23,09,03,06,19,17,22,01,,,,,2.4,1.3,2.0*36
$GPRMC,181100.000,A,3941.9565,N,00031.9254,W,0.00,347.43,290817,,,A*7F
$GPGGA,181101.000,3941.9565,N,00031.9254,W,1,08,1.3,266.9,M,51.9,M,,0000*42
$GPGSA,A,3,23,09,06,03,19,17,22,01,,,,,2.4,1.3,2.0*36
$GPRMC,181101.000,A,3941.9565,N,00031.9254,W,0.00,347.43,290817,,,A*7E
$GPGGA,181102.000,3941.9565,N,00031.9254,W,1,08,1.3,266.9,M,51.9,M,,0000*41
$GPGSA,A,3,23,09,06,03,19,17,22,01,,,,,2.4,1.3,2.0*36
$GPGSV,3,1,12,23,74,055,31,09,72,222,37,06,45,311,20,03,44,070,30*73
$GPGSV,3,2,12,19,38,257,32,17,32,231,37,22,24,084,25,01,20,139,26*7B
$GPGSV,3,3,12,07,13,167,17,38,00,000,,02,05,316,,11,02,153,*74
$GPRMC,181102.000,A,3941.9565,N,00031.9254,W,0.00,347.43,290817,,,A*7D
$GPGGA,181103.000,3941.9565,N,00031.9254,W,1,08,1.3,266.9,M,51.9,M,,0000*40
$GPGSA,A,3,23,09,06,03,19,17,22,01,,,,,2.4,1.3,2.0*36
$GPRMC,181103.000,A,3941.9565,N,00031.9254,W,0.00,347.43,290817,,,A*7C
```

En el monitor serie de arduino también aparece las mismas tramas,con lo que se transmite correctamente la información y a velocidad adecuada.

4º PASO

Ahora que ya se transmiten correctamente las tramas NMEA recibidas por el gps a Matlab la información que necesita está dentro de esas tramas separadas por comas. Por lo que se tendrá que utilizar la técnica denominada parseo para obtener la información que se necesita de la trama que la contiene. Por ejemplo: la trama '$GPGGA' es la que contiene latitud, longitud, altura y la '$GPRMC' contiene velocidad sobre la superficie de la tierra y la hora y fecha.

Por lo que solo necesitaremos que nos mande información de las tramas '$GPGGA'y '$GPRMC' por lo que las demás las excluimos. Para realizar esto creamos una condición para que si reconoce los 5 primeros dígitos de la trama son iguales a los que se necesitan que lo represente sino que la descarte.

```
if(strncmpi(data,'$GPGGA',6))==1
if(strncmpi(data,'$GPRMC',6))==1
```

De este modo solo sale información de estas dos tramas que son las que se necesitan.
Más tarde se parsea delimitando por comas ya que entre la comas está cada información
Por ejemplo: la velocidad se encuentra entre las comas nº 7 y 8 de la trama $GPRMC con lo que se extrae y se almacena en un vector para almacenar cada segundo que muestrea el valor de la velocidad que al estar en nudos se tendrá que convertir a km/h.

$GPRMC,184027.000,A,3941.9573,N,00031.9247,W,0.00,347.43,290817,,,A*7B

```
ii=findstr(data,',');
vel=data(ii(7)+1:ii(8)-1);
disp(vel)
```

Y lo mismo se realiza para obtener latitud, longitud y altura, pero en la latitud y longitud se tendrá que modificar el signo positivo o negativo dependiendo en que hemisferio se encuentre ya que la información se entrega en grados y minutos por lo que se transformarán a decimal.

$GPGGA,184028.000,3941.9573,N,00031.9247,W,1,10,0.9,273.3,M,51.9,M,,0000*44

```
if (ns == 'S')
     lat = -lat;
      end

if (ew == 'W')
      lon=-lon;
        end
latitud = floor(lat/100)+rem(lat,100)/60;
longitud = floor(lon/100)+rem(lon,100)/60;
%conversion de grados y minutos a decimales
```

Una vez realizadas estas modificaciones guardábamos los datos en vectores para luego representarlos en Matlab mediante la API de google maps. Realizando una prueba para obtener el recorrido que se hubiese trazado mientras estuviese ejecutándose el programa con los siguientes resultados:

Ilustración 19. Trazado realizado con gps

Ilustración 20. zoom trazado de ruta

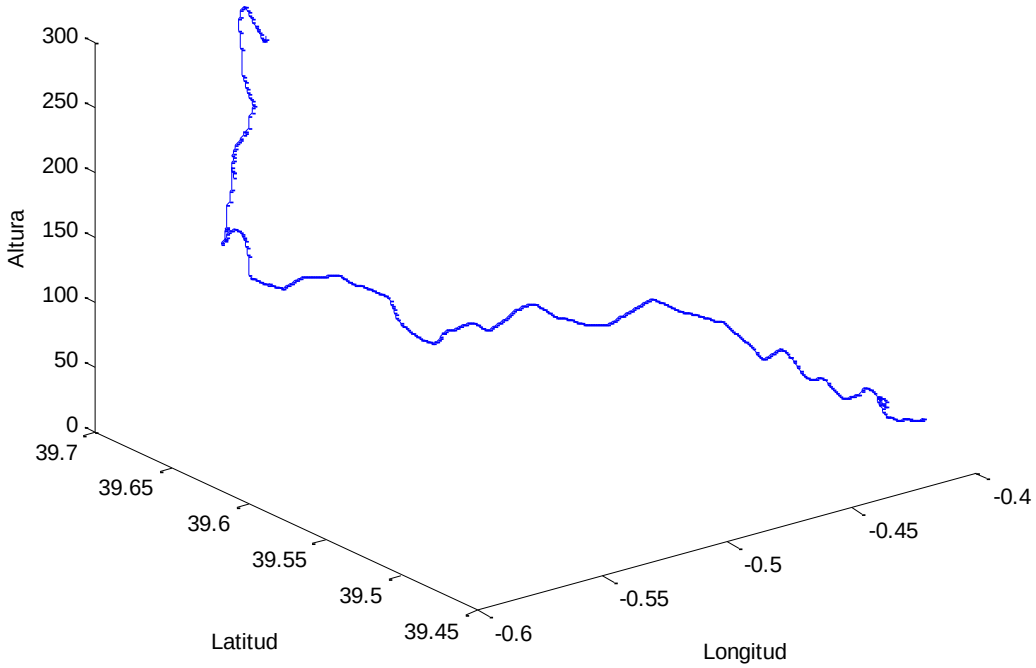


Ilustración 21. .Trayecto en 3D

Támbien se puede obtener la altura a medida que se hace el recorrido, representarla e incluso obtener el valor mínimo y máximo y lo mismo con la velocidad.

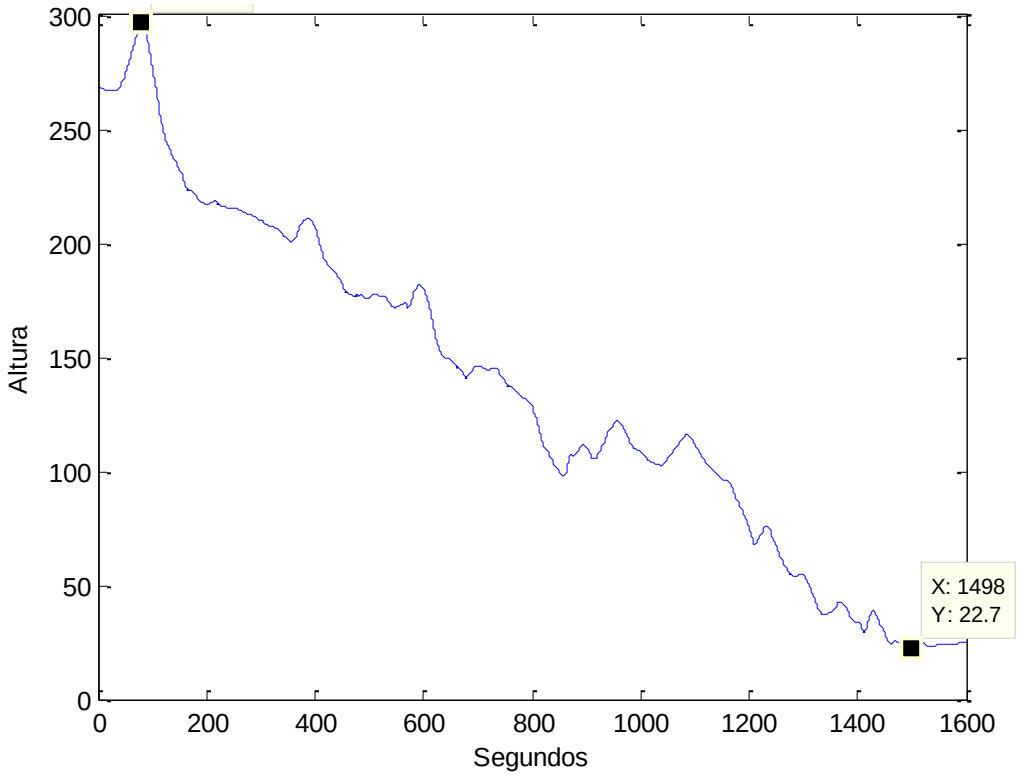


Ilustración 22.. Altura del trayecto con valor mín y máx

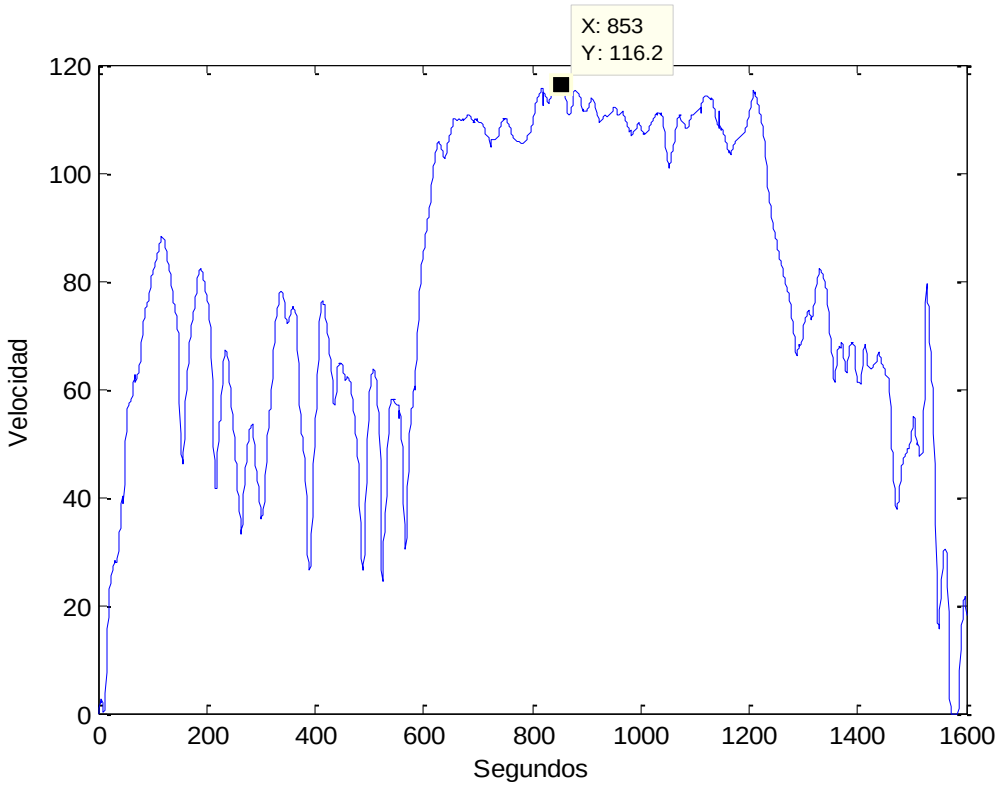


Ilustración 23.. Velocidad de trayecto con valor máx

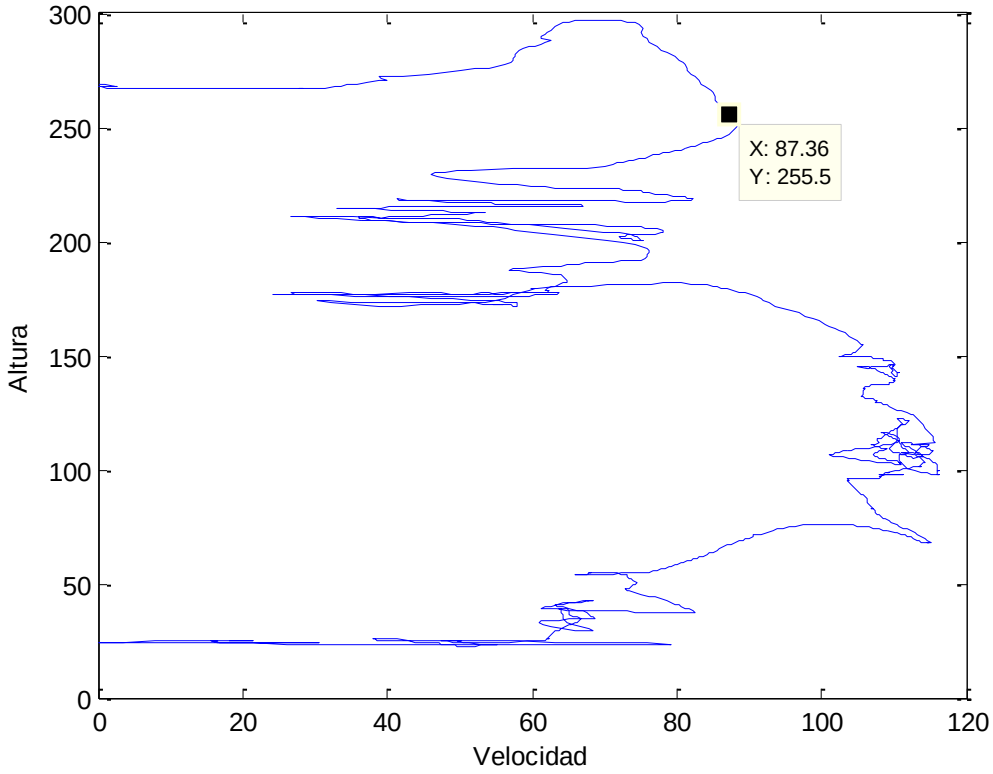


Ilustración 24. Velocidad y Altura

Velocidad media de trayecto: 75.8024 km/h.

5º PASO

Se crea una interfaz gráfica en Matlab para que se realice mejor la interacción y puesta en marcha de la aplicación por el usuario. Con gráficas de altura, velocidad y la posición de latitud y longitud sobre un mapa. Con botón de inicio/parada. A continuación el siguiente código de implementación de ésta.

```
lngmin=-0.8 ; lngmax=-0.4 ; latmin=39.3 ; latmax=39.8 ; T=1 ; L=2000 ;
lnglmn=-0.55 ; lnglmx=-0.45 ; latlmn=39.4 ; latlmx=39.6 ;

fig=figure ;
set(fig,'Color',[0.9 0.9
0.9],'Name','GPS','Units','normalized','Position',[0.01 0.05 0.98
0.9]) ;
set(fig,'MenuBar','none','ToolBar','figure') ;
aa1=axes('position',[0.5 0.2 0.45 0.7],'Color',[1 1 1],'Box','on') ;
pp0=plot(0,0,'o','LineWidth',1,'MarkerSize',10,'MarkerFaceColor',[1 1
1]) ; grid on ; hold on ;
pp1=plot(zeros(1,L),zeros(1,L),'.k') ;
pp3=plot([lnglmn lnglmx lnglmx lnglmn lnglmn],[latlmx latlmx latlmn
latlmn latlmx],'g','Linewidth',2) ;

bbMS=uicontrol('Style','togglebutton','Units','normalized') ;
set(bbMS,'Position',[0.6 0.05 0.25 0.1],'String','Mapa/Satelite') ;
set(bbMS,'Callback','if (get(gco,''Value''))
plot_google_map(''MapType'',''satellite'') ; else
plot_google_map(''MapType'',''route'') ; end ;') ;

axis([lngmin lngmax latmin latmax])
```

```
plot_google_map('Refresh',1,'AutoAxis',1) ;

aa2=axes('position',[0.05 0.7 0.4 0.2],'Color',[1 1 1],'Box','on') ;
pp4=plot(0,0,'r.','MarkerSize',5) ; grid on ; hold on ; axis([0 1600 0
150]) ;
tt1=uicontrol('Style','text','Units','normalized','ForegroundColor',[1
0 0],'BackgroundColor',[0.9 0.9 0.9],'FontSize',12) ;
set(tt1,'Position',[0.05 0.6 0.4 0.05],'String',sprintf('Velocidad =
%2.2f km/h',0)) ;

aa3=axes('position',[0.05 0.35 0.4 0.2],'Color',[1 1 1],'Box','on') ;
pp5=plot(0,0,'b.','MarkerSize',5) ; grid on ; hold on ; axis([0 1600 0
300]) ;
tt2=uicontrol('Style','text','Units','normalized','ForegroundColor',[0
0 1],'BackgroundColor',[0.9 0.9 0.9],'FontSize',12) ;
set(tt2,'Position',[0.05 0.25 0.4 0.05],'String',sprintf('Altitud =
%2.2f m',0)) ;

axes(aa1) ;

bbSS=uicontrol('Style','togglebutton','Units','normalized') ;
set(bbSS,'Position',[0.1 0.05 0.25 0.1],'String','Start/Stop') ;
set(bbSS,'Callback','if (get(gco,''Value'')) stop=0 ; else stop=1 ;
end ;') ;
```

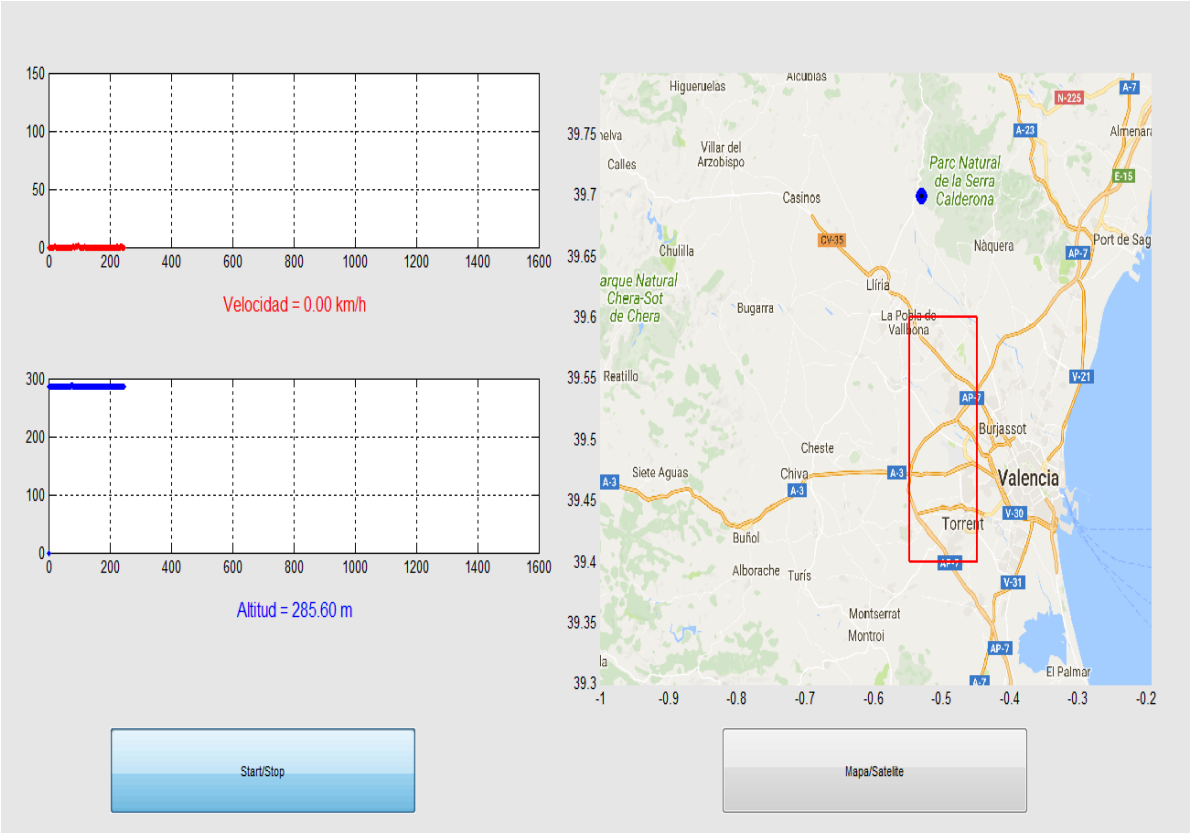
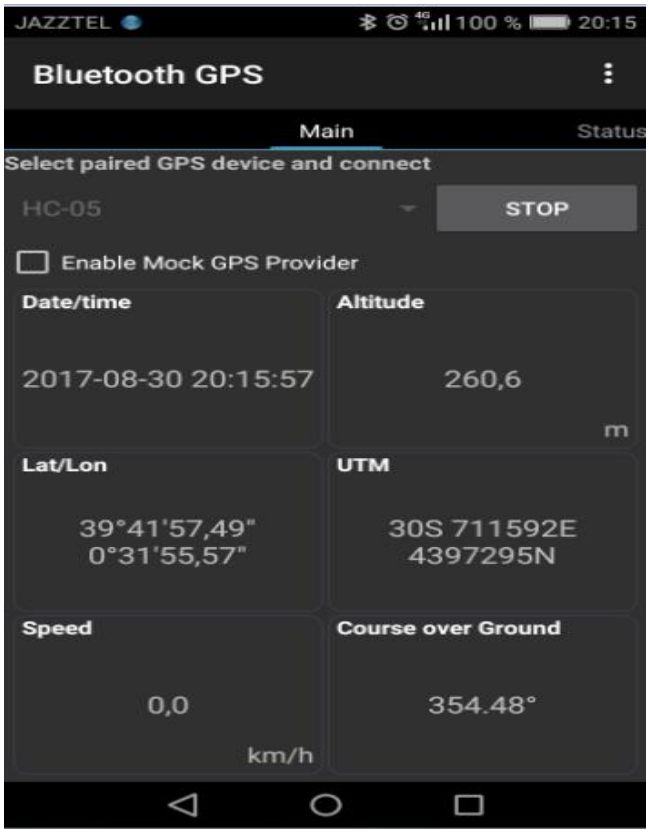
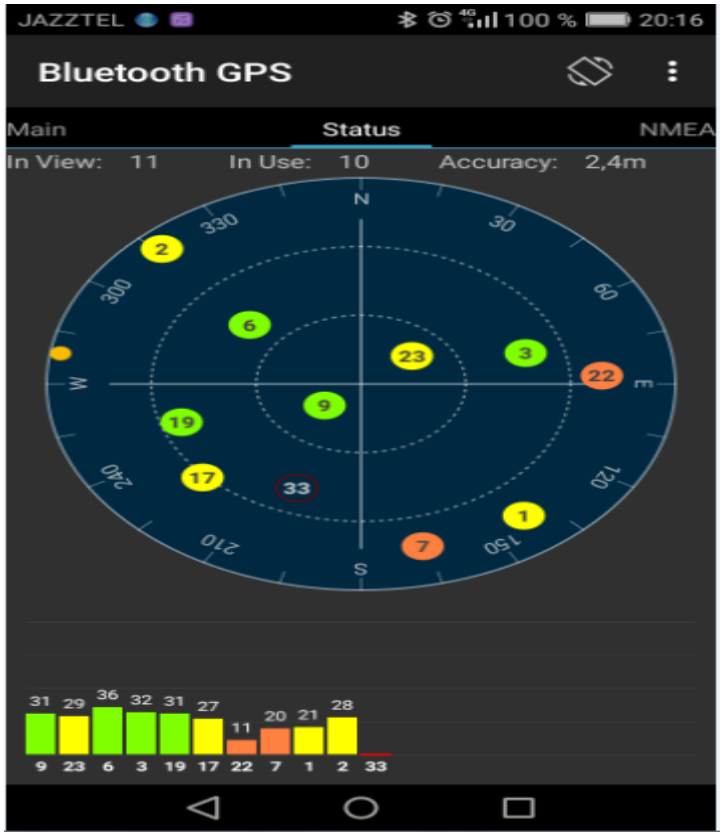


Ilustración 25. Interfaz usuario Matlab

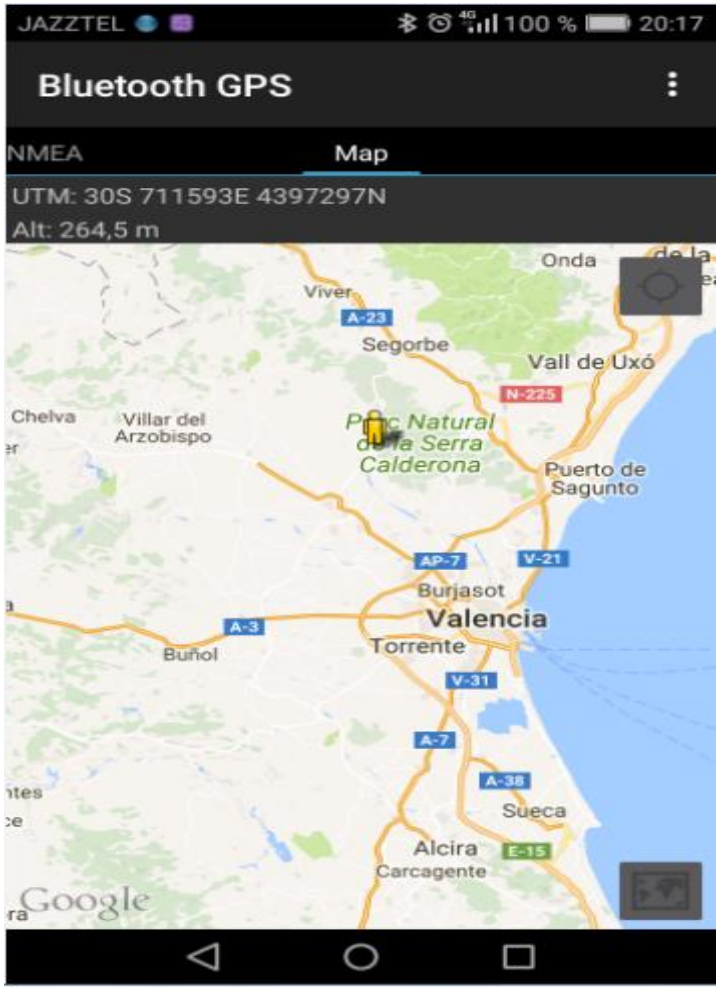
Se han realizado varias pruebas para determinar el correcto funcionamiento de la interfaz y la recepción de datos en un tiempo óptimo, validándola y comprobando que todas las etapas de transmisión, recepción y procesado son correctas y funciona en tiempo real.

También existe la posibilidad de conectar el módulo bluetooth HC-05 a un Smartphone y que obtenga los diferentes datos transmitidos, para esto será necesario descargar la aplicación Bluetooth GPS y se obtendrán los siguientes parámetros como se muestran en las siguientes imágenes.






JAZZTEL
100 %
20:16
Bluetooth GPS
Status
NMEA
Map
CLEAR
PAUSE
$GPGGA,181613.000,3941.9585,N,00031.9258,W,1,10,0.8,260.1,M,
51.9,M,,0000*49
$GPGSA,A,3,09,23,06,03,19,17,22,07,01,02,,,1.5,0.8,1.2*3A
$GPRMC,181613.000,A,3941.9585,N,00031.9258,W,
0.00,354.48,300817,,,A*79
$GPGGA,181614.000,3941.9585,N,00031.9258,W,1,10,0.8,260.1,M,
51.9,M,,0000*4E
$GPGSA,A,3,09,23,06,03,19,17,22,07,01,02,,,1.5,0.8,1.2*3A
$GPRMC,181614.000,A,3941.9585,N,00031.9258,W,
0.00,354.48,300817,,,A*7E
$GPGGA,181615.000,3941.9585,N,00031.9258,W,1,10,0.8,260.1,M,
51.9,M,,0000*4F
$GPGSA,A,3,09,23,06,03,19,17,22,07,01,02,,,1.5,0.8,1.2*3A
$GPRMC,181615.000,A,3941.9585,N,00031.9258,W,
0.00,354.48,300817,,,A*7F
$GPGGA,181616.000,3941.9586,N,00031.9258,W,1,10,0.8,260.1,M,
51.9,M,,0000*4F
$GPGSA,A,3,09,23,06,03,19,17,22,07,01,02,,,1.5,0.8,1.2*3A
$GPRMC,181616.000,A,3941.9586,N,00031.9258,W,
0.00,354.48,300817,,,A*7F
$GPGGA,181617.000,3941.9586,N,00031.9258,W,1,10,0.8,260.1,M,
51.9,M,,0000*4E
$GPGSA,A,3,09,23,06,03,19,17,22,07,01,02,,,1.5,0.8,1.2*3A
$GPGSV,
3,1,11,09,76,228,33,23,70,050,31,06,49,309,36,03,41,074,32*7A
$GPGSV,
3,2,11,19,36,252,31,17,29,228,30,22,21,087,17,07,17,166,22*7E
$GPGSV,3,3,11,01,16,141,19,02,08,316,28,33,41,202,*42


JAZZTEL
100 %
20:17
Bluetooth GPS
NMEA
Map
UTM: 30S 711593E 4397297N
Alt: 264,5 m
Onda
Viver
A-23
Segorbe
Vall de Uxó
N-225
Chelva
Villar del
Arzobispo
Parc Natural
de la Serra
Calderona
Puerto de
Sagunto
AP-7
V-21
Burjasot
Valencia
A-3
Buñol
Torrente
V-31
A-7
A-38
Sueca
Alcira
E-15
Carcagente
Google

6. BIBLIOGRAFÍA

www.google.com

www.mathworks.com

www.arduiniana.org

7. CONCLUSIONES

En este proyecto se ha desarrollado el procedimiento de montaje de un kit GPS para mostrar la capacidad que tienen los microcontroladores como arduino y que con un sensor GPS tan pequeño y sin antena externa es capaz de recibir datos de satélites que se encuentran en la órbita terrestre y la precisión que se alcanza de márgenes de error entre 1 y 10 metros.

Estos sensores a día de hoy se están instalando en infinidad de aplicaciones por ejemplo: drones(aviones no tripulados) ,smartphones con los que podemos acceder al GPS que nos proporcionan y conocer en que posicionamiento estamos y si queremos realizar algún trazado, que dirección tomar.

También están presentes en relojes inteligentes que utilizan senderistas o corredores y ver los diferentes parámetros que puede mostrar este aparato, para ayudarles en su entrenamiento o para localizar vehículos robados en fin, en infinidad de aplicaciones se podría utilizar este sensor y cada vez más con la nuevas tecnologías.

Pero no todo es bueno también puede hacerse con efecto negativo por pérdida de privacidad ya que se puede utilizar para controlar y hacer seguimiento de gente sin su consentimiento y conocer donde se encuentra y que trayectos ha realizado, pero como tiene más beneficios positivos que negativos será una herramienta indispensable en un futuro próximo.

Para ampliar este proyecto se realizaría una app con interfaz de usuario híbrida para Android e IOS por lo que se podría utilizar en aparatos más reducidos de tamaño que un PC.

ANEXO 1

En el anexo se van a mostrar los códigos de programación utilizados para el desarrollo del proyecto así como las características del GPS.

dato_gps §

```
#include <SoftwareSerial.h>
SoftwareSerial gps(2,3);  //crear puerto digital
// SoftwareSerial bt(4,5);
char dato=' ';    //crear variable dato

void setup()
{
 Serial.begin(4800);       //velocidad puerto serie
 gps.begin(4800);         //velocidad puerto digital
 //bt.begin(4800);
}


void loop()
{
  if(gps.available())       //si encuentra datos que los guarde en la variable dato
  {
    dato=gps.read();
    Serial.write(dato);   //mostrar datos en el puerto serie
  }
}
```

CODIGO_gps2

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
TinyGPSPlus gps;
SoftwareSerial ss(2,3);

void setup() {
   Serial.begin(4800);
 ss.begin(4800);
}
void loop() {

while (ss.available() > 0)
  gps.encode(ss.read());
if (gps.altitude.isUpdated())
{
Serial.print("A=");Serial.println(gps.altitude.meters());
Serial.print("LAT="); Serial.println(gps.location.lat(), 6);
Serial.print("LON="); Serial.println(gps.location.lng(), 6);
//Serial.print("Satélites disponibles= ");Serial.println(gps.satellites.value());
Serial.print("V=");Serial.println(gps.speed.kmph());
delay(1000);
}

}
// end loop()
```

Código en Matlab 1

```
clc
clear all
close all
test =Bluetooth('HC-05',1);
fopen(test);                %abrir conexión Bluetooth
i=1;
LATITUD=zeros(1,end);       % crea matriz para guardar datos de
latitud
LONGITUD=zeros(1,end);      % crea matriz para guardar datos de
longitud
ALTITUD=zeros(1,end);       % crea matriz para guardar datos de
altitud
VELOCIDAD=zeros(1,end);     % crea matriz para guardar datos de
velocidad
TIME=zeros(1,end);          % crea matriz para guardar datos de tiempo

Vmedia=median(VELOCIDAD);   % velocidad media
while 1;
    data=fscanf(test);          %escanear puerto serie
    if(strncmpi(data,'$GPRMC',6))==1    %si coincide con las 6 letras
de la trama mostrar sino descartar
         ii=findstr(data,',');          %encontrar comas
          fecha=data(ii(9)+1:ii(10)-1);  %posición entre comas del
dato de fecha dentro de la trama
          dia = str2num(fecha(1:2));  % seleccionar los 2 primeros
términos de cadena y pasarlo a numeros
            mes = str2num(fecha(3:4)); % seleccionar los 2 segundos
términos de cadena y pasarlo a numeros
           year = str2num(fecha(5:end)); % seleccionar los últimos
términos de cadena y pasarlo a numeros
            Fecha =
strcat(num2str(dia),'/',num2str(mes),'/',num2str(year)); %mostrar
fecha formato digital
           disp(Fecha)                  %mostrar fecha
         vel=data(ii(7)+1:ii(8)-1);     %posición entre comas del dato
de velocidad dentro de la trama
         vel=str2num(vel);              %convertir cadena a número
         vel=vel*1.852;             %convertir nudos a km/h
         disp(vel)              %mostrar velocidad
        VELOCIDAD(i)=vel(1);    %guardar datos de velocidad en una
matriz de 1 fila
    end
    if(strncmpi(data,'$GPGGA',6))==1    %si coincide con las 6 letras
de la trama mostrar sino descartar
%         disp(data)
       ii=findstr(data,',');             %encontrar comas
         time=data(ii(1)+1:ii(2)-1);    % %posición entre comas del
dato de tiempo dentro de la trama
            hour = str2num(time(1:2));  % seleccionar los 2 primeros
términos de cadena y pasarlo a numeros
            hour=hour+2;                %ajustar hora
            minute = str2num(time(3:4)); % Convertir minutos a
segundos
            second = str2num(time(5:end)); % cargar segundos
           Hora =
strcat(num2str(hour),':',num2str(minute),':',num2str(second));
%mostrar hora digital
            time=str2num(time);         %convertir cadena a número
```

```
            time=time+20000;              %ajustar hora
              TIME(i)=time(1);            %guardar datos de tiempo en
una matriz de 1 fila
          disp(Hora)                      %mostrar tiempo
          ns = data(ii(3)+1);         % norte o sur
          lat=data(ii(2)+1:ii(3)-1);    %posición entre comas del dato
de latitud dentro de la trama
          lat=str2num(lat);             %convertir cadena a número
          lat = floor(lat/100)+rem(lat,100)/60;     %conversion de
grados y minutos a decimales
          if (ns == 'S')                    % si es sur la latitud es
negativa
          lat = -lat;
          end
          disp(lat)                   %mostrar latitud
          LATITUD(i)=lat(1);          %guardar datos de latitud en una
matriz de 1 fila

        ew = data(ii(5)+1);           %oeste o este
        lon=data(ii(4)+1:ii(5)-1);  %posición entre comas del dato de
longitud dentro de la trama
        lon=str2num(lon);             %convertir cadena a número
        lon = floor(lon/100)+rem(lon,100)/60;  %conversion de grados y
minutos a decimales
        if (ew == 'W')      %si es este la longitud es negativa
            lon=-lon;
        end
            disp(lon)             %mostrar longitud
            LONGITUD(i)=lon(1); %guardar datos de longitud en una
matriz de 1 fila

        alt=data(ii(9)+1:ii(10)-1);  %posición entre comas del dato de
altura dentro de la trama
        alt=str2double(alt);          %convertir cadena a número
        disp(alt)                   %mostrar altura
        ALTITUD(i)=alt(1);          %guardar datos de longitud en una
matriz de 1 fila
       i=i+1;
    end

end
 plot_google_map                %abrir google maps
 plot(LONGITUD,LATITUD,'.r','MarkerSize',20) %dibujar latitud y
longitud en un mapa de google maps

fclose(test);
delete(test);
clear (test);
```

Código en Matlab con interfaz

```
test =Bluetooth('HC-05',1);
fopen(test);

lngmin=-0.8 ; lngmax=-0.4 ; latmin=39.3 ; latmax=39.8 ; T=1 ; N=1 ;
L=200 ;
lnglmn=-0.55 ; lnglmx=-0.45 ; latlmn=39.4 ; latlmx=39.6 ;

fig=figure ;
```

```
set(fig,'Color',[0.9 0.9
0.9],'Name','GPS','Units','normalized','Position',[0.01 0.05 0.98
0.9]) ;
set(fig,'MenuBar','none','ToolBar','figure') ;
aa1=axes('position',[0.5 0.2 0.45 0.7],'Color',[1 1 1],'Box','on') ;
pp0=plot(0,0,'o','LineWidth',1,'MarkerSize',10,'MarkerFaceColor',[1 1
1]) ; grid on ; hold on ;
pp1=plot(zeros(1,L),zeros(1,L),'.k') ;
pp3=plot([lnglmn lnglmx lnglmx lnglmn lnglmn],[latlmx latlmx latlmn
latlmn latlmx],'g','Linewidth',2) ;

bbMS=uicontrol('Style','togglebutton','Units','normalized') ;
set(bbMS,'Position',[0.6 0.05 0.25 0.1],'String','Mapa/Satelite') ;
set(bbMS,'Callback','if (get(gco,''Value''))
plot_google_map(''MapType'',''satellite'') ; else
plot_google_map(''MapType'',''route'') ; end ;') ;

axis([lngmin lngmax latmin latmax])
plot_google_map('Refresh',1,'AutoAxis',1) ;

aa2=axes('position',[0.05 0.7 0.4 0.2],'Color',[1 1 1],'Box','on') ;
pp4=plot(0,0,'r.','MarkerSize',5) ; grid on ; hold on ; axis([0 1600 0
150]) ;
tt1=uicontrol('Style','text','Units','normalized','ForegroundColor',[1
0 0],'BackgroundColor',[0.9 0.9 0.9],'FontSize',12) ;
set(tt1,'Position',[0.05 0.6 0.4 0.05],'String',sprintf('Velocidad =
%2.2f km/h',0)) ;

aa3=axes('position',[0.05 0.35 0.4 0.2],'Color',[1 1 1],'Box','on') ;
pp5=plot(0,0,'b.','MarkerSize',5) ; grid on ; hold on ; axis([0 1600 0
300]) ;
tt2=uicontrol('Style','text','Units','normalized','ForegroundColor',[0
0 1],'BackgroundColor',[0.9 0.9 0.9],'FontSize',12) ;
set(tt2,'Position',[0.05 0.25 0.4 0.05],'String',sprintf('Altitud =
%2.2f m',0)) ;

axes(aa1) ;

bbSS=uicontrol('Style','togglebutton','Units','normalized') ;
set(bbSS,'Position',[0.1 0.05 0.25 0.1],'String','Start/Stop') ;
set(bbSS,'Callback','if (get(gco,''Value'')) stop=0 ; else stop=1 ;
end ;') ;

stop=1 ;
while (stop)
    pause(T) ;
end


Y=[];       % crea matriz para guardar datos de latitud
X=[];      % crea matriz para guardar datos de longitud
A=[];       % crea matriz para guardar datos de altitud
V=[];     % crea matriz para guardar datos de velocidad
x=0;
y=0;
v=0;
a=0;
iii=0 ;
```

```
while (stop==0)
    iii=iii+1 ;
    data=fscanf(test);                %escanear puerto serie
         if(strncmpi(data,'$GPRMC',6))==1    %si coincide con las 6
letras de la trama mostrar sino descartar
         ii=findstr(data,',');          %encontrar comas
y=lati(N*ii) ;
         vel=data(ii(7)+1:ii(8)-1);     %posición entre comas del dato
de velocidad dentro de la trama
         vel=str2double(vel);              %convertir cadena a número
         vel=vel*1.852;     %convertir nudos a km/h
         v=vel      %cambiar nombre variable
v=vel(N*ii) ;
         if v>=0;
             V=[V v];      %guardar datos de velocidad en una matriz
          end

         end
             if(strncmpi(data,'$GPGGA',6))==1    %si coincide con las
6 letras de la trama mostrar sino descartar
        ii=findstr(data,',');             %encontrar comas
        ns = data(ii(3)+1);       % norte o sur
          lat=data(ii(2)+1:ii(3)-1);    %posición entre comas del dato
de latitud dentro de la trama
          lat=str2double(lat);              %convertir cadena a número
          lat = floor(lat/100)+rem(lat,100)/60;     %conversion de
grados y minutos a decimales
          if (ns == 'S')          % si es sur la latitud es negativa
          lat = -lat;
          end
           y=lat    %cambiar nombre variable
           Y=[Y y];  %guardar datos de latitud en una matriz

        ew = data(ii(5)+1);         %oeste o este
        lon=data(ii(4)+1:ii(5)-1);  %posición entre comas del dato de
longitud dentro de la trama
        lon=str2double(lon);           %convertir cadena a número
        lon = floor(lon/100)+rem(lon,100)/60;  %conversion de grados y
minutos a decimales
        if (ew == 'W')      %si es este la longitud es negativa
            lon=-lon;
        end

           x=lon        %cambiar nombre variable
           X=[X x]; %guardar datos de longitud en una matriz

        alt=data(ii(9)+1:ii(10)-1);  %posición entre comas del dato de
altura dentro de la trama
        alt=str2double(alt);         %convertir cadena a número
        a=alt           %cambiar nombre variable
%        a=alt(N*ii) ;
         A=[A a];   %guardar datos de longitud en una matriz
             end

   [x y v a]    % mostrar variables por pantalla
%   x=longi(N*ii) ;
%               y=lati(N*ii) ;
%                           v=vel(N*ii) ;
%     a=alt(N*ii) ;
```

```
    axes(aa2) ;
    V=get(pp4,'YData') ; V=[V v] ;
    set(pp4,'XData',[0:N*T:N*T*iii],'YData',V) ;
    A=get(pp5,'YData') ; A=[A a] ;
    set(pp5,'XData',[0:N*T:N*T*iii],'YData',A) ;

    axes(aa1) ;
    set(pp0,'XData',x,'YData',y) ;
    set(pp0,'MarkerFaceColor',[v/120 0 1-(v/120)]) ;
    X=get(pp1,'XData') ; X=[x X(1:end-1)] ;
    Y=get(pp1,'YData') ; Y=[y Y(1:end-1)] ;
    set(pp1,'XData',X,'YData',Y) ;
    if (x>lnglmx || x<lnglmn || y>latlmx || y<latlmn)
        set(pp3,'Color',[1 0 0]) ;
    else
        set(pp3,'Color',[0 1 0]) ;
    end

    set(tt1,'String',sprintf('Velocidad = %2.2f km/h',v)) ; %mostrar
velocidad en interfaz
    set(tt2,'String',sprintf('Altitud = %2.2f m',a)) ;  %mostrar
altitud en interfaz
    pause(T/N) ;
end
```

Código en Matlab 2

```
simulado=1 ;

if (simulado)
    load recorrido.mat
    N=10 ;
else
    test =Bluetooth('HC-05',1);
    fopen(test);
end

lngmin=-0.8 ; lngmax=-0.4 ; latmin=39.3 ; latmax=39.8 ; T=1 ; L=2000 ;
lnglmn=-0.55 ; lnglmx=-0.45 ; latlmn=39.4 ; latlmx=39.6 ;

fig=figure ;
set(fig,'Color',[0.9 0.9
0.9],'Name','GPS','Units','normalized','Position',[0.01 0.05 0.98
0.9]) ;
set(fig,'MenuBar','none','ToolBar','figure') ;
aa1=axes('position',[0.5 0.2 0.45 0.7],'Color',[1 1 1],'Box','on') ;
pp0=plot(0,0,'o','LineWidth',1,'MarkerSize',10,'MarkerFaceColor',[1 1
1]) ; grid on ; hold on ;
pp1=plot(zeros(1,L),zeros(1,L),'.k') ;
pp3=plot([lnglmn lnglmx lnglmx lnglmn lnglmn],[latlmx latlmx latlmn
latlmn latlmx],'g','Linewidth',2) ;

bbMS=uicontrol('Style','togglebutton','Units','normalized') ;
set(bbMS,'Position',[0.6 0.05 0.25 0.1],'String','Mapa/Satelite') ;
```

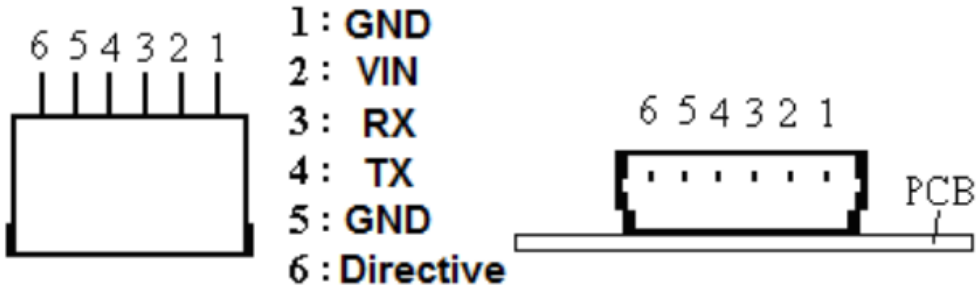
```
set(bbMS,'Callback','if (get(gco,''Value''))
plot_google_map(''MapType'',''satellite'') ; else
plot_google_map(''MapType'',''route'') ; end ;') ;

axis([lngmin lngmax latmin latmax])
plot_google_map('Refresh',1,'AutoAxis',1) ;

aa2=axes('position',[0.05 0.7 0.4 0.2],'Color',[1 1 1],'Box','on') ;
pp4=plot(0,0,'r.','MarkerSize',5) ; grid on ; hold on ; axis([0 1600 0
150]) ;
tt1=uicontrol('Style','text','Units','normalized','ForegroundColor',[1
0 0],'BackgroundColor',[0.9 0.9 0.9],'FontSize',12) ;
set(tt1,'Position',[0.05 0.6 0.4 0.05],'String',sprintf('Velocidad =
%2.2f km/h',0)) ;

aa3=axes('position',[0.05 0.35 0.4 0.2],'Color',[1 1 1],'Box','on') ;
pp5=plot(0,0,'b.','MarkerSize',5) ; grid on ; hold on ; axis([0 1600 0
300]) ;
tt2=uicontrol('Style','text','Units','normalized','ForegroundColor',[0
0 1],'BackgroundColor',[0.9 0.9 0.9],'FontSize',12) ;
set(tt2,'Position',[0.05 0.25 0.4 0.05],'String',sprintf('Altitud =
%2.2f m',0)) ;

axes(aa1) ;

bbSS=uicontrol('Style','togglebutton','Units','normalized') ;
set(bbSS,'Position',[0.1 0.05 0.25 0.1],'String','Start/Stop') ;
set(bbSS,'Callback','if (get(gco,''Value'')) stop=0 ; else stop=1 ;
end ;') ;

stop=1 ;
while (stop)
    pause(T) ;
end
Y=[];       % crea matriz para guardar datos de latitud
X=[];      % crea matriz para guardar datos de longitud
A=[];       % crea matriz para guardar datos de altitud
V=[];       % crea matriz para guardar datos de velocidad
x=0;
y=0;
v=0;
a=0;
iii=0 ;

while (stop==0)

    if (simulado)
        iii=iii+10 ;
        x=longi(iii) ;
        y=lati(iii) ;
        v=vel(iii) ;
        a=alti(iii) ;
    else
        iii=iii+1 ;
         data=fscanf(test);     % escanear puerto serie
         disp(data)
         N=1;
    if(strncmpi(data,'A',1))==1    %si coincide con las 1 letras de la
trama mostrar sino descartar
```

```
        ii=findstr(data,'=');       % encontrar signo =
        a=data(ii(1)+1:end);          %encontrar dato después de
signo igual
        a=str2double(a);              %convertir variable de cadena
a número
        A=[A a];                      %almacenar en vector
    end
    if(strncmpi(data,'LAT',3))==1    %si coincide con las 3 letras de
la trama mostrar sino descartar
        ii=findstr(data,'=');         % encontrar signo =
        y=data(ii(1)+1:end);          %encontrar dato después de
signo igual
        y=str2double(y);              %convertir variable de cadena
a número
        Y=[Y y];                      %almacenar en vector
    end
    if(strncmpi(data,'LON',3))==1    %si coincide con las 3 letras de
la trama mostrar sino descartar
        ii=findstr(data,'=');       % encontrar signo =
        x=data(ii(1)+1:end);        %encontrar dato después de signo
igual
        x=str2double(x);            %convertir variable de cadena a
número
        X=[X x];                    %almacenar en vector
    end
    if(strncmpi(data,'V',1))==1    %si coincide con las 1 letras de la
trama mostrar sino descartar
        ii=findstr(data,'=');       % encontrar signo =
        v=data(ii(1)+1:end);        %encontrar dato después de signo
igual
        v=str2double(v);            %convertir variable de cadena a
número
        V=[V v];                    %almacenar en vector
    end
    end
    [x y v a]    % mostrar variables por pantalla

    axes(aa2) ;
     V=get(pp4,'YData') ; V=[V v];
     set(pp4,'XData',[0:N*T:T*iii],'YData',V) ;
     axes(aa3) ;
     A=get(pp5,'YData') ; A=[A a] ;
     set(pp5,'XData',[0:N*T:T*iii],'YData',A) ;

     axes(aa1) ;
     set(pp0,'XData',x,'YData',y) ;
     set(pp0,'MarkerFaceColor',[0 1 1]);
     X=get(pp1,'XData') ; X=[x X(1:end-1)] ;
     Y=get(pp1,'YData') ; Y=[y Y(1:end-1)] ;
     set(pp1,'XData',X,'YData',Y) ;
     if (x>lnglmx || x<lnglmn || y>latlmx || y<latlmn)
         set(pp3,'Color',[1 0 0]) ;
     else
         set(pp3,'Color',[0 1 0]) ;
     end

     set(tt1,'String',sprintf('Velocidad = %2.2f km/h',v)) ; %mostrar
velocidad en interfaz
     set(tt2,'String',sprintf('Altitud = %2.2f m',a)) ;  %mostrar
altitud en interfaz
```

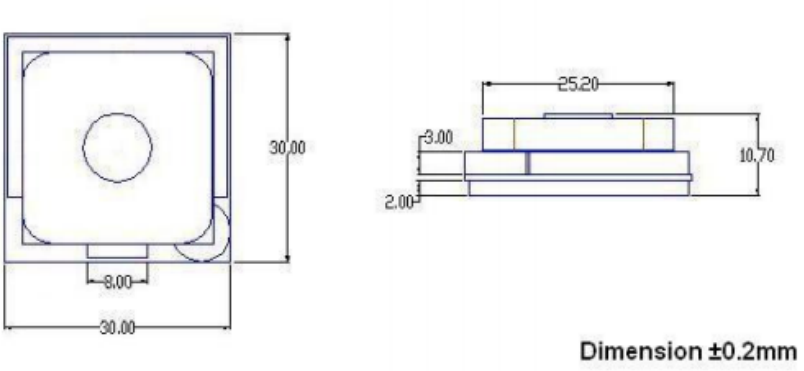
```
    if (simulado)
     pause(T/N) ;
    end
end
```

GPS EM-506

Product Pin Description



PIN Number(s)	Name	Type	Description	Note
1,5	GND	P	Ground.	
2	VIN	P	This is the main power supply to the engine board. (4.5Vdc to 6.5Vdc)	
3	RXD	I	This is the main receive channel for receiving software commands to the engine board from SiRFdemo software or from user written software. Baud rate based on flash memory setting.	
4	TXD	O	This is the main transmits channel for outputting navigation and measurement data to user's navigation software or user written software. Output 3.3V level.	
6	Directive	O	This pin indicates the GPS states.	



OPERATING Description

GND

This is Ground pin for the baseband circuit.

VIN

This is the main power supply to the engine board. (4.5Vdc to 6.5Vdc)

RXD

This is the main channel for receiving software commands from SiRFdemo software or from your proprietary software.

TXD

This is the main transmits channel for outputting navigation and measurement data to user's navigation software or user written software. Output is TTL level, 0V ~ 3.3V.

PLIEGO DE CONDICIONES

En primer lugar se debe tener en cuenta las especificaciones de diseño que debemos cumplir:

Tensión continua de entrada 5V.

Temperatura admisible en el circuito 70 ºC.

Humedad relativa del 0% al 90%.

El prototipo desarrollado está preparado para trabajar mediante una placa Arduino con las siguientes características:

Arduino Uno.

Tensión aplicada al dispositivo 5V

Frecuencia de funcionamiento de Arduino 16 MHz.

Número de pines digitales 54 (15 de ellos proveen salida PWM).

16 pines analógicos.

Memoria Flash de 256KB.

Memoria Sram de 8KB.

Memoria EEprom de 4KB.

Velocidad de reloj de 16 MHz.

Corriente DC por cada Pin Entrada/Salida: 40 mA.

Corriente DC entregada en el pin 3.3V: 50 mA.

INTRODUCCIÓN

El objetivo del siguiente pliego de condiciones es la ordenación de las condiciones técnicas y facultativas que han de estar presentes en la ejecución de este proyecto.

Este proyecto va a ser supervisado por un ingeniero de sistemas, que en este caso va a ser el profesor encargado de dirigir el proyecto.

El pliego de prescripciones técnicas establece la definición del montaje en cuanto a su naturaleza intrínseca.

1.2 CONDICIONES TÉCNICAS

Las siguientes condiciones técnicas serán de obligado cumplimiento para el técnico de obra, el cual deberá aceptarlas y comprometerse a acatarlas. Todos los trabajos serán realizados por personas especialmente preparadas, tanto en el conexionado de la placa como en las demás partes que comprenden el proyecto.

Corresponde al técnico de obra:

Redactar los complementos o rectificadores que se precisen.

Redactar cuando sea requerido el estudio de los sistemas adecuados a los puestos de trabajo en la realización de la obra y aprobar el Plan de Seguridad y Salud para la aplicación del mismo.

Comprobar las instalaciones provisionales, medios auxiliares y sistemas de seguridad e higiene en el trabajo, controlando su correcta ejecución.

1.2.2 CONDICIONES DE MONTAJE

El director de montaje así como el director de diseño de programación quedan obligados a que todas las dudas que surjan de la interpretación de los elementos del proyecto o posteriormente durante su uso serán resueltas por la dirección facultativa de acuerdo con el Pliego de Condiciones Técnicas. Cuando se trate de aclarar, interpretar o modificar preceptos de los pliegos de condiciones o indicaciones de los planos o croquis, las órdenes o instrucciones correspondientes se comunicarán precisamente por escrito al instalador de obra, estando este obligado a devolver los originales o las copias suscribiendo con su firma el enterado, que figurará al pie de todas las órdenes, avisos e instrucciones que reciba el técnico de obra. El instalador podrá requerir del Técnico de Obra según sus respectivos cometidos, las instrucciones o aclaraciones que precisen para la correcta interpretación y ejecución del proyecto.

Las especificaciones no descritas en el siguiente pliego con relación al proyecto, y que figuren en el resto de documentación (memoria, planos y presupuesto) deben considerarse como dato a tener en cuenta a la formulación del presupuesto.

Los componentes serán reconocidos antes de su montaje sin cuya aprobación no podránser utilizados. Existirá el derecho de ser cosechados aquellos componentes que no reúnan las condiciones en base a sus características técnicas para ser considerados a punto.

1.3 CONDICIONES ECONÓMICAS

Todos los precios están sujetos a variaciones, ya que su valor no es constante sino quevaría a través del tiempo.

Los componentes acopiados, una vez abonados por el propietario, son de la exclusiva propiedad de éste, de su guarda y protección será encargado el contratista.

Los pagos se efectuarán por el propietario en los plazos previamente establecidos, y su importe, corresponderá precisamente al de las certificaciones de ejecución del proyecto, en virtud de las cuales se verifican aquellos.

En los precios estarán siempre incluido lo siguiente:

Suministro de equipos y materiales eléctricos principales indicados, materiales

consumibles y materiales auxiliares.

Costos de mano de obra directa e indirecta, costos de maquinaria, mantenimiento de

instalaciones temporales, desmantelamiento de instalaciones temporales, gastos generales, costos del cumplimiento de las Normas y Reglamento de Seguridad e

Higiene necesarios para la protección y seguridad del personal durante la realización del trabajo.

1.4 CONDICIONES TÉCNICAS

El trabajo se realizará de acuerdo con la legislación vigente y las especificaciones técnicas, en general, atendiendo especialmente al reglamento electrotécnico para baja tensión e instrucciones técnicas complementarias. Todos los componentes utilizados serán de primera calidad y reunirán las condiciones exigidas en el Reglamento Electrotécnico de baja tensión y demás disposiciones vigentes referidas a materiales. Todos los materiales podrán ser sometido a análisis o pruebas, por cuenta de la contrata para asegurar su calidad. Cualquier otro que haya sido especificado y sea necesario emplear deberá ser aprobado por la Dirección Técnica, bien entendiendo que será rechazado el que no reúna las condiciones exigidas, por la buena práctica del proyecto.

Los materiales no consignados en el proyecto que dieran lugar a precios contradictorios

reunirán las condiciones de bondad necesarias, a juicio de la Dirección Facultativa, no teniendo el contratista derecho a reclamación alguna por estas condiciones exigidas.

Todos y cada uno de los equipos principales a suministrar dispondrán en un lugar visible una placa característica, en la que estarán grabados de forma indeleble los siguientes datos:

Nombre del fabricante o marca comercial.

Designación del tipo.

Número de serie de fabricación.

Grado de protección mecánica.

Características técnicas.

PRESUPUESTO

A continuación se detallan los componentes necesarios para la construcción del Kit. Piezas necesarias que deben ser introducidas para una correcta puesta en funcionamiento y construcción conforme ha sido diseñado.

CONCEPTO	UDS	CANTIDAD	PRECIO	IMPORTE
A00R36 Placa arduino uno	U	1	16.5	16.5
S0225 Sparkfun GPS Starter Kit	U	1	50	50
P4T456 Módulo bluetooth HC-05	U	1	7.5	7.5
SR32 Cable usb	U	1	3.30	3.30
FH225 Conjunto de 10 cables	U	1	2.1	2.10
VERT67 Pila 9V	U	1	2.10	2.5
Montar componentes y programar con código	H	1	25	25

IMPORTE BRUTO	IVA(21%)	IMPORTE TOTAL €
106.9	22.45	129.35