

29-9-2022

MATERIA:
SENSORES Y
ACTUADORES.
PROFESORES:
MORALES, JORGE-
VERA, GONZALO

Tadeo Enrique Zarate

EJERCICIO 1:

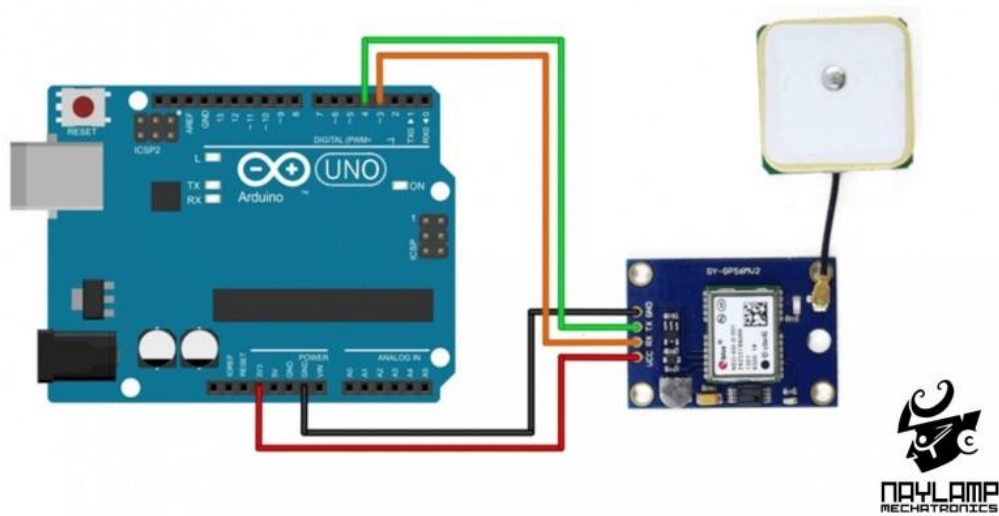
F) COMO IMPLEMENTARIA UN SENSOR INTELIGENTE DE POSICION GLOBAL.

El **Sistema de Posicionamiento Global (GPS)**; En ingles, ***Global Positioning System***), originalmente **Navstar GPS**, es un sistema que permite localizar cualquier objeto (una persona, un vehículo, etc.) sobre la Tierra con una precisión de hasta centímetros (si se utiliza GPS diferencial), aunque lo común son unos pocos metros.



IMPLEMENTACION

El módulo GPS en su modelo GY-GPS6MV2 viene con un módulo de serie U-Blox NEO 6M equipado en el PCB, una EEPROM con configuración de fábrica, una pila de botón para mantener los datos de configuración en la memoria EEPROM, un indicador LED y una antena cerámica. También posee los pines o conectores Vcc, Rx, Tx y Gnd por el que se puede conectar a algún microcontrolador mediante una interfaz serial. Para que nuestro módulo GPS funcione a la perfección se recomienda hacer las pruebas en un ambiente abierto o cercano a la ventana para una correcta recepción de la señal.



Bien, ahora vamos a probar nuestro módulo conectándolo a nuestro Arduino (en este caso se usará un Arduino UNO) mediante un puerto serie que se emulará por Software ya que usaremos el Rx0 y TX0 (puerto serie por Hardware) para la comunicación con nuestra PC y así verificar los datos que recibimos por el módulo GPS.

Vamos a realizar las conexiones que se muestran en la siguiente imagen o seguir los pasos que se describen a continuación:

- Conecte el pin 3.3V del Arduino UNO al pin Vcc del módulo GPS.
- Conecte el pin GND del Arduino UNO al pin GND del módulo GPS.
- Conecte el pin digital 4 del Arduino UNO al pin Tx del módulo GPS.
- Conecte el pin digital 3 del Arduino UNO al pin Rx del módulo GPS.

Cabe indicar que nuestro módulo GPS también se puede alimentar con una tensión de 5V ya que posee un regulador integrado dentro de sí.

A continuación, cargaremos el siguiente código a nuestro Arduino, en el cual se puede apreciar que se emplea la librería *SoftwareSerial* para emular un puerto serie como se mencionó anteriormente (no es necesaria instalarla, ya que viene por defecto en nuestro IDE de Arduino).

CODIGO

```
#include <SoftwareSerial.h>

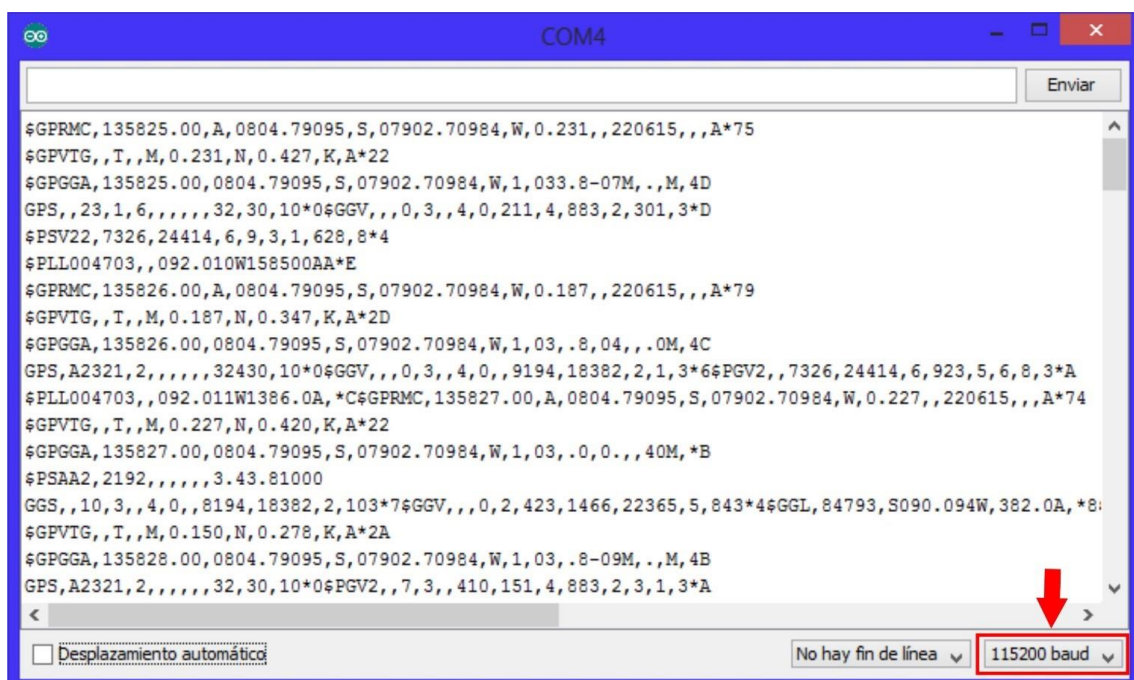
SoftwareSerial gps(4,3);

char dato=' ';

void setup()
{
  Serial.begin(115200);
  gps.begin(9600);
}

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

Como podemos ver, lo que hace nuestro programa es leer constantemente el módulo GPS a una velocidad de 9600 baudios que es la velocidad por la que viene configurado por defecto nuestro módulo GPS y enviar dichos datos a la PC a través del puerto serie físico para poder visualizarlos en nuestro Monitor



Serial. Al abrir nuestro Monitor Serial, nos aseguramos de configurarlo a una velocidad de 115200 baudios. Podremos ver los datos que recibimos.

Los datos que recibimos en nuestro módulo GPS siguen el protocolo NMEA (siglas de National Marine Electronics Association), las cuales son sentencias estándares para la recepción de datos GPS. Una de ellas y la más usada son las sentencias \$GPRMC, las cuales tienen la siguiente estructura:

\$GPRMC,044235.000,A,4322.0289,N,00824.5210,W,0.39,65.46,020615,,,A*44

Donde si analizamos la trama de este ejemplo y basándose en el protocolo NMEA, podríamos determinar las siguientes variables:

- 044235.000** representa la hora GMT (04:42:35)
- **"A"** es la indicación de que el dato de posición está fijado y es correcto. "V" sería no válido
- 4322.0289** representa la longitud (43° 22.0289')
- N** representa el Norte
- 00824.5210** representa la latitud (8° 24.5210')
- W** representa el Oeste
- 0.39** representa la velocidad en nudos
- 65.46** representa la orientación en grados
- 020615** representa la fecha (2 de Junio del 2015)

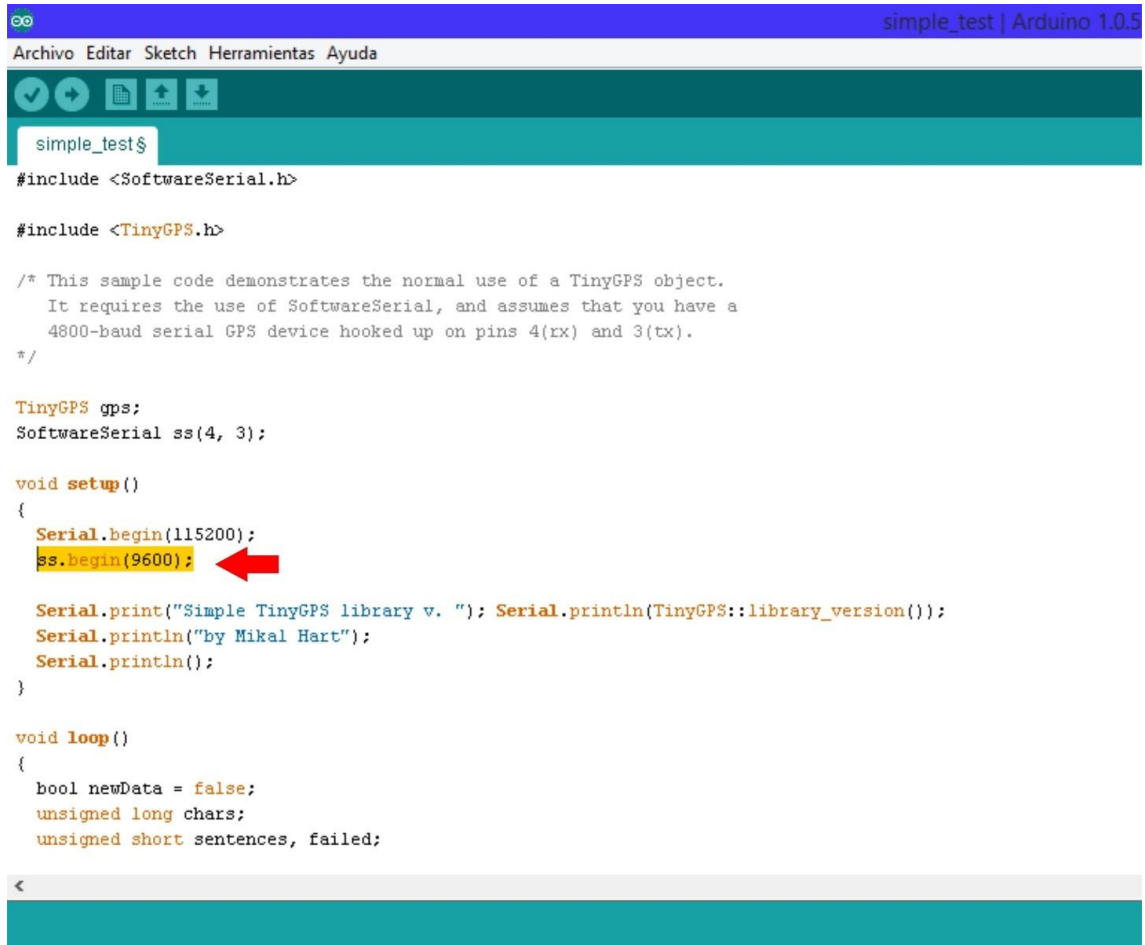
Como vimos, de la trama de datos que nos envía nuestro módulo GPS podemos obtener varias variables, siendo las importantes para proyectos de posicionamiento la latitud y la longitud. Para ello, vamos a hacer uso de la librería TinyGPS que la podemos descargar de aquí:

<https://github.com/mikalhart/TinyGPS>

Recuerde que una vez descargada la Librería, tenemos que importarla copiándola en la carpeta "Libraries" donde se instaló nuestro IDE de Arduino y luego reiniciar el programa para que sea cargada correctamente. La librería TinyGPS nos facilitará la identificación tanto de la latitud y longitud, así como las otras variables descritas anteriormente sin tener que recurrir a algoritmos complejos para lograr obtenerlas. Para ello ejecutamos un sencillo ejemplo que nos provee la librería, para lo cual nos vamos a **Archivo/Ejemplos/TinyGPS/simple_test** en nuestro IDE de Arduino.

A continuación, nos aseguramos de cambiar la velocidad de lectura del puerto

serie emulado a 9600 baudios y cargamos nuestro código a nuestra tarjeta Arduino.



```
simple_test | Arduino 1.0.5
Archivo Editar Sketch Herramientas Ayuda

simple_test$
#include <SoftwareSerial.h>

#include <TinyGPS.h>

/* This sample code demonstrates the normal use of a TinyGPS object.
   It requires the use of SoftwareSerial, and assumes that you have a
   4800-baud serial GPS device hooked up on pins 4(rx) and 3(tx).
*/

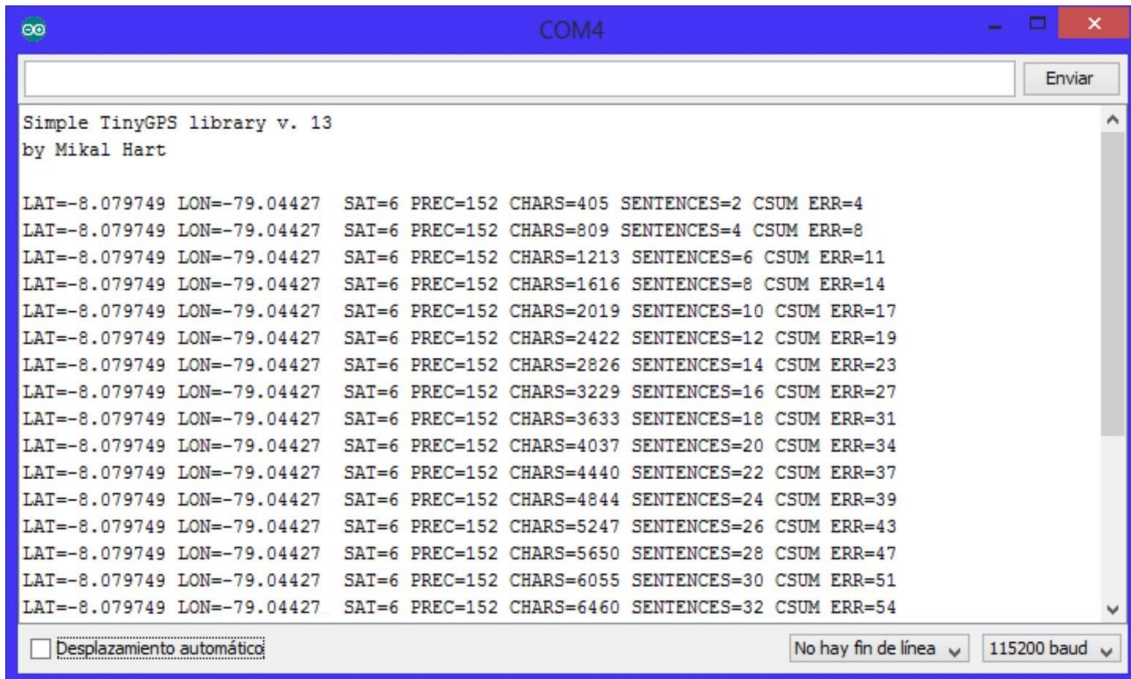
TinyGPS gps;
SoftwareSerial ss(4, 3);

void setup()
{
  Serial.begin(115200);
  ss.begin(9600);

  Serial.print("Simple TinyGPS library v. "); Serial.println(TinyGPS::library_version());
  Serial.println("by Mikal Hart");
  Serial.println();
}

void loop()
{
  bool newData = false;
  unsigned long chars;
  unsigned short sentences, failed;
```

Finalmente, podremos abrir nuestro Monitor serial y visualizar las variables mencionadas tal como se pueden ver en la siguiente imagen:



COM4

Enviar

Simple TinyGPS library v. 13
by Mikal Hart

```
LAT=-8.079749 LON=-79.04427 SAT=6 PREC=152 CHARS=405 SENTENCES=2 CSUM ERR=4
LAT=-8.079749 LON=-79.04427 SAT=6 PREC=152 CHARS=809 SENTENCES=4 CSUM ERR=8
LAT=-8.079749 LON=-79.04427 SAT=6 PREC=152 CHARS=1213 SENTENCES=6 CSUM ERR=11
LAT=-8.079749 LON=-79.04427 SAT=6 PREC=152 CHARS=1616 SENTENCES=8 CSUM ERR=14
LAT=-8.079749 LON=-79.04427 SAT=6 PREC=152 CHARS=2019 SENTENCES=10 CSUM ERR=17
LAT=-8.079749 LON=-79.04427 SAT=6 PREC=152 CHARS=2422 SENTENCES=12 CSUM ERR=19
LAT=-8.079749 LON=-79.04427 SAT=6 PREC=152 CHARS=2826 SENTENCES=14 CSUM ERR=23
LAT=-8.079749 LON=-79.04427 SAT=6 PREC=152 CHARS=3229 SENTENCES=16 CSUM ERR=27
LAT=-8.079749 LON=-79.04427 SAT=6 PREC=152 CHARS=3633 SENTENCES=18 CSUM ERR=31
LAT=-8.079749 LON=-79.04427 SAT=6 PREC=152 CHARS=4037 SENTENCES=20 CSUM ERR=34
LAT=-8.079749 LON=-79.04427 SAT=6 PREC=152 CHARS=4440 SENTENCES=22 CSUM ERR=37
LAT=-8.079749 LON=-79.04427 SAT=6 PREC=152 CHARS=4844 SENTENCES=24 CSUM ERR=39
LAT=-8.079749 LON=-79.04427 SAT=6 PREC=152 CHARS=5247 SENTENCES=26 CSUM ERR=43
LAT=-8.079749 LON=-79.04427 SAT=6 PREC=152 CHARS=5650 SENTENCES=28 CSUM ERR=47
LAT=-8.079749 LON=-79.04427 SAT=6 PREC=152 CHARS=6055 SENTENCES=30 CSUM ERR=51
LAT=-8.079749 LON=-79.04427 SAT=6 PREC=152 CHARS=6460 SENTENCES=32 CSUM ERR=54
```

☐ Desplazamiento automático

No hay fin de línea

115200 baud