**Explicación Firmware**:

Se ha desarrollado con un módulo GPS uBox-neo6 debido a problemas de logística para conseguir el modelo de GPS solicitado. Al ser las tramas NMEA un standard no debe haber problema con la ejecución del codigo.

He desarrollado la prueba en una tarjeta de desarrollo Nucleo-F091RC, que tiene un uC STM32F091, el IDE utilizado es el ST Cube IDE versión 1.6.1.

A pesar que pedían una solución simple, quise desarrollarlo con FreeRTOS en 4 tareas para demostrar mis capacidades de desarrollo en firmware.

El firmware se compone de 4 tareas:

* **GPS**: Tarea encargada de recibir por puerto serial las tramas NMEA y decodificar
* **Shell**: Tarea encargada de recibir y configurar comandos
* **CheckPos**: tarea encargada de verificar la distancia entre el punto configurado y la posición actual.
* **WDTCheck**: Tarea encargada de vigilar que las otras tareas no estén bloqueadas y de activar un led de supervivencia. Tiene una pequeña interfaz para cambiar la velocidad de parpadeo del led para indicar eventos, en este caso se utiliza para mostrar la cercanía al punto.

A continuacion voy a describir cada una de las tareas:

**Tarea GPS:** Decidí hacer la librería GPS porque tengo experiencia y conocimiento en la manipulación de tramas NMEA. Utilicé esta pagina para extraer los campos de la trama RMC, que es donde viene latitud y longitud, además de la hora y fecha:

http://aprs.gids.nl/nmea/#rmc

Una vez recibida la trama NMEA, verifico que sea la trama RMC, calculo el CRC y verifico que los campos de las tramas estén en un rango válido, por ejemplo que la latitud este entre -90 y 90, que el campo de advertencia de receptor de navegación (**navigation receiver warning)** sea el carácter “A”, entre otros. Segundo a segundos voy haciendo esta extracción de datos y si son válidos, los paso a una estructura permanente. Adicionalmente capturo el PPS para determinar que los datos ya de la estructura permanente no están actualizados. Hay una flag que indica que si los datos son válidos o no.

**Tarea Shell:** A traves de un puerto serial a 215300 bps se encarga de recibir y configurar los siguientes comandos:

|  |  |
| --- | --- |
| **Comando** | **Explicación** |
| lat=*value* | Configura la latitud del punto en grados decimales |
| lon=*value* | Configura la longitud del punto en grados decimales |
| rad=*value* | Configura el radio de la tierra en kilometros |
| Data | Muestra los valores configurados de la siguiente manera:  Point:[4.754822 -74.090868] Radius:6378.100000 Kms |

**Tarea CheckPos:** cada 5 segundos, solicita a la tarea gps la latitud y la longitud y mediante la ecuación de Haversine calcula la distancia entre 2 puntos. La formula la saqué de esta página:

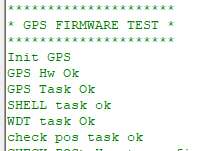
https://www.geeksforgeeks.org/haversine-formula-to-find-distance-between-two-points-on-a-sphere/

Debido a que la tierra no es totalmente esférica, se debe reajustar el radio de la tierra de acuerdo a la ubicación geográfica.

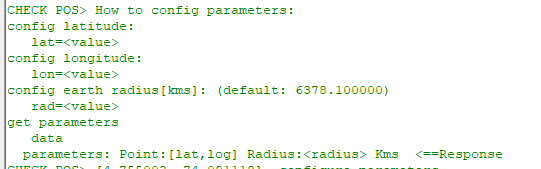
Configura el parpadeo del led para indicar si se encuentra cerca del punto. Configura a menos de 20, 50, 100, 250, 500 y mas de 500 metros.

**Tarea WDT\_Check:** Cada 10 segundos hace un heart-bit a las demás tareas y espera la confirmación, en caso de no obtener respuesta reinicia el sistema. Otra responsabilidad que tiene es el de estar refrescando el watchdog timer. Si esta tarea se bloquea, el watchdog timer reinicia el sistema. Finalmente se encarga de hacer parpadear un led para indicar que el sistema se encuentra en ejecución.

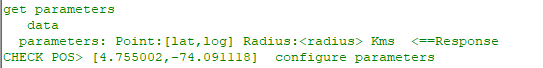
Cuando inicia el sistema informa el inicio de las tareas



Muestra la ayuda de los comandos



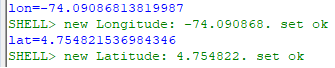
Muestra la posición que el GPS que tiene en este momento, en caso de no tener configurado la latitud y la longitud indica que debe configurarse



Si recibe algo que no es un comando lo ignora e informa



cuando se configura informa si el comando fue aceptado o no



En caso de un valor no válido no acepta la configuración:



Una vez configurado indica la distancia que hay entre el punto actual y el punto configurado.



Cuando esta a menos de 100 metros lo indica:



informa si los datos del GPS no son válidos



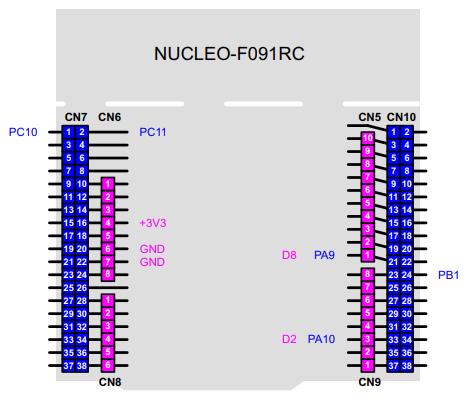
Una vez los datos del GPS son válidos, vuelve hacer los cálculos



**CONEXIÓN ELECTRICA:**

De la tarjeta Nucleo-F091RC se utilizaron los siguientes pines:

https://www.st.com/resource/en/user\_manual/um1724-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf

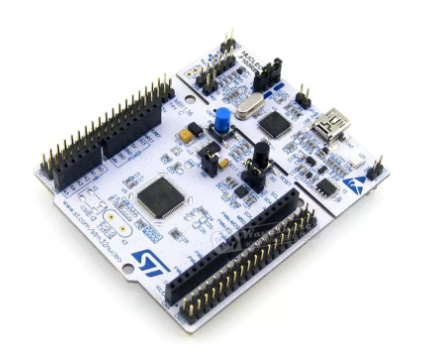
****

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **NUCLEO-F091RC** | | | | **MODULO GPS UBLOX NEO 6M** | |
| **Conector** | **Puerto** | **Pin uc** | **Funcion** | **Funcion** | **Pin** |
| CN10-33 | PA10 | 43 | Rx uC | Tx GPS | TxD |
| CN10-21 | PA9 | 42 | Tx uC | Rx GPS | RxD |
| CN10-24 | PB1 |  | Irq PPS | PPS Signal | PPS |
| CN6-4 | n.a. | n.a. | +3.3V | Vcc In | VCC |
| CN6-6 | n.a. | n.a. | GND | Ground | GND |

Para el puerto serie de la Shell se utilizó un cable interfaz FTDI TTL-232RG

https://co.mouser.com/datasheet/2/163/DS\_TTL-232RG\_CABLES-6645.pdf

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **NUCLEO-F091RC** | | | | **Interfaz FTDI TTL-232RG** | |
| **Conector** | **Puerto** | **Pin uc** | **Funcion** | **Funcion** | **Color** |
| CN7-2 | PC11 | 52 | Rx uC | Tx GPS | Amarillo |
| CN7-1 | PC10 | 53 | Tx uC | Rx GPS | Naranja |
| CN6-7 | n.a. | n.a. | GND | Ground | GND |

****

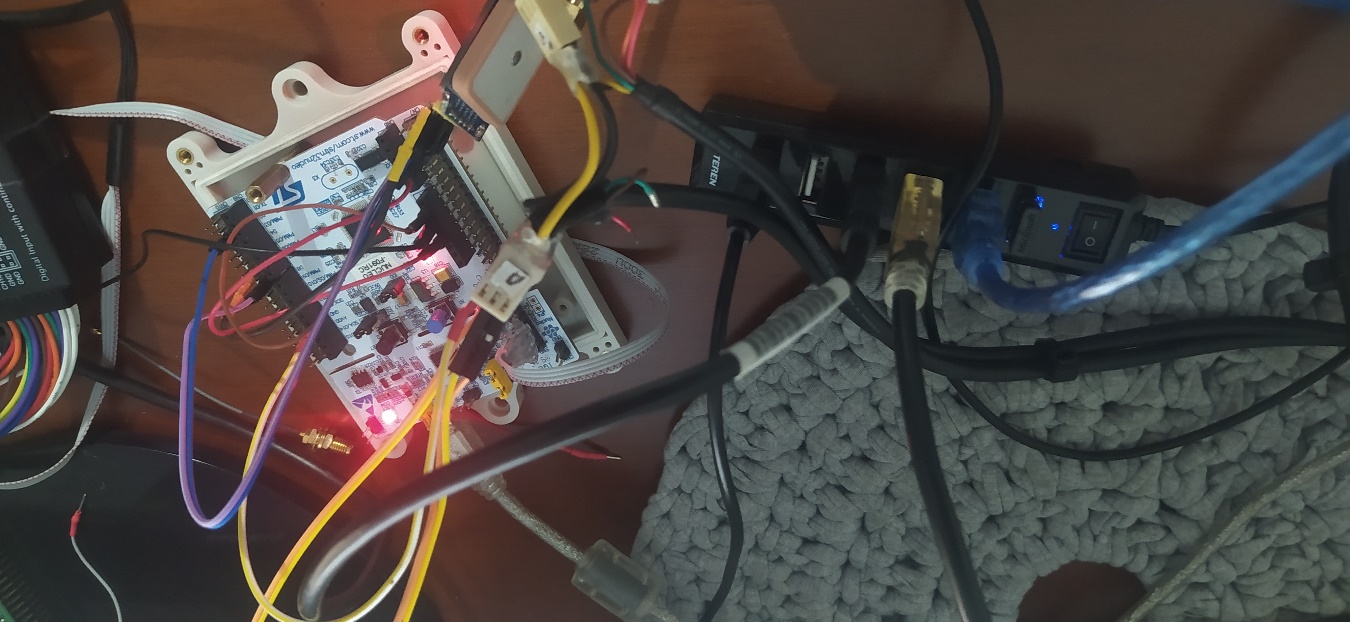
**Tarjeta Nucleo-F091RC**



**Modulo GPS ublox NEO 6m**



**Cable FTDI TTL-232RG**



**Montaje**