FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS UNIVERSIDAD DE CHILE
CENTRO DE ENERGÍA

PROYECTO CIL
CENTRO DE INNOVACIÓN DEL LITIO

Prueba de Comunicación TINET

Versión 1.0

Este documento se divide en dos secciones que describen la comunicación vía Bluetooh 4.0 entre un prototipo de desarrollo y una aplicación de prueba y la documentación del Firmware utilizado. La primera parte muestra los pasos a seguir para la implementación del Firmware dentro de la tarjeta electrónica y la instalación de la aplicación dentro de un dispositivo móvil. Mientras que la segunda se encarga de explicar de forma sencilla el Firmware para entregar a los usuarios un manejo claro del código. Tanto el Firmware como la aplicación de prueba se encuentran en el siguiente link.

Parte 1: Implementación Firmware y Aplicación

Firmware

En este documento no se detalla el contenido de la tarjeta de desarrollo o prototipo, sino que el objetivo de éste es mostrar al lector como compilar y cargar el código a la placa. Basta con señalar que el prototipo cuenta con un módulo Bluetooh, un módulo GPS, un adaptador de tarjeta SD, un sensor de corriente, un sensor de voltaje, un sensor de temperatura y una placa Arduino Mega Pro 2560 3.3v.

En la carpeta de *Dropbox -> Elibatt-Arduino -> libraries* están todas las librerías utilizadas para el proyecto. Las librerías correspondientes al sensor de temperatura, GPS, lectura de sensores, etc.

Para compilar el archivo *pruebaTinet.ino*, el IDE de Arduino debe apuntar al directorio donde se encuentren las librerias, por ejemplo: *C:\Users\Name-User\Documents\Github\E-liBatt-master\Firmware*

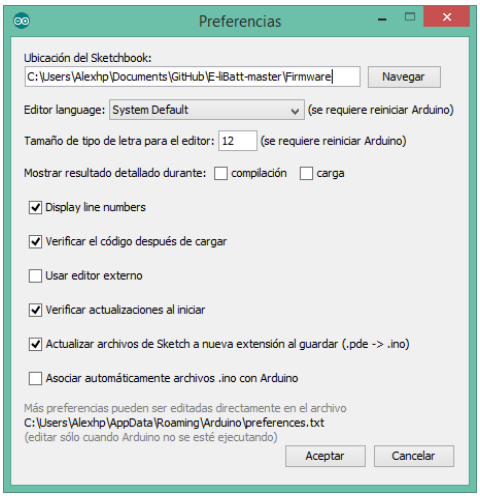

Figura 1. Path Arduino

Si se ha seteado bien la carpeta de *Sketchbook*, debiese verse tal como muestra la **Figura 2**. Con las librerías del GPS "Adafruit_GPS", del sensor de temperatura "MCP9808", etc.

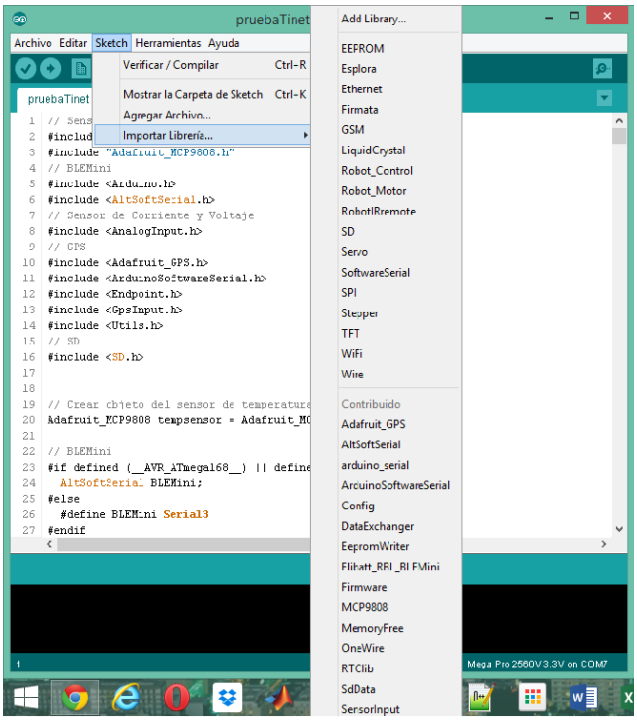

Figura 2. Librerías agregadas

Con esto ya es posible compilar. El proceso de compilación se visualiza en la **Figura 3**.

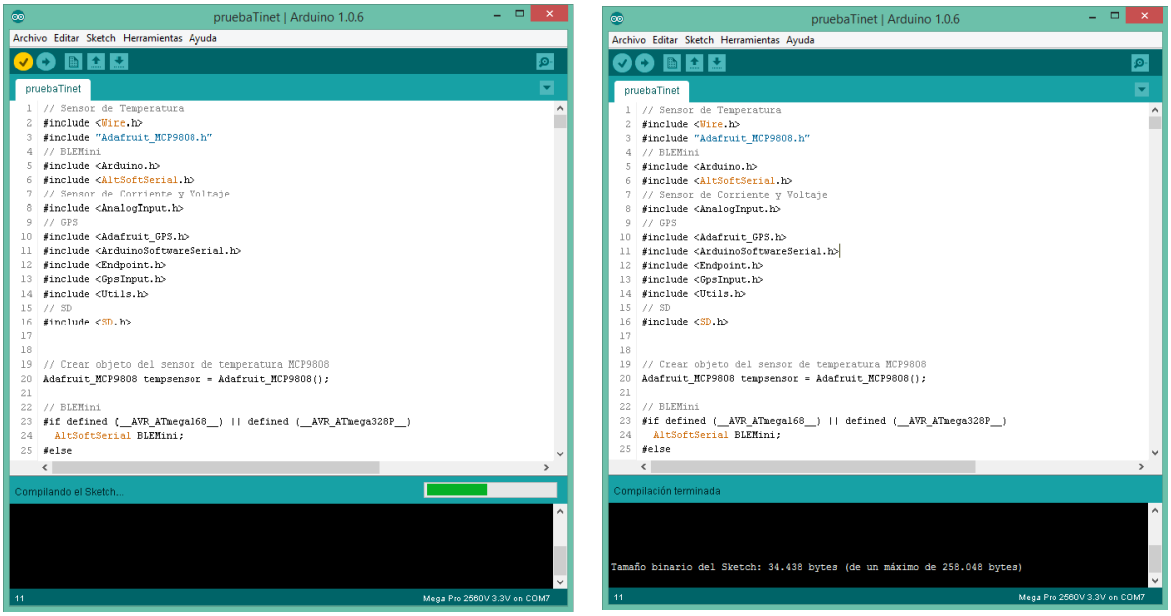

Figura 3. Compilar código

Siguiendo estos pasos, es posible cargar el archivo a la placa mediante el botón *Cargar* mostrado en la **Figura 4**.

Figura 4. Cargar código

Por último, mediante el *Monitor Serial* que proporciona el IDE de Arduino es posible verificar su funcionamiento a través de *Herramientas -> Monitor Serial*, tal como muestra la **Figura 5**.

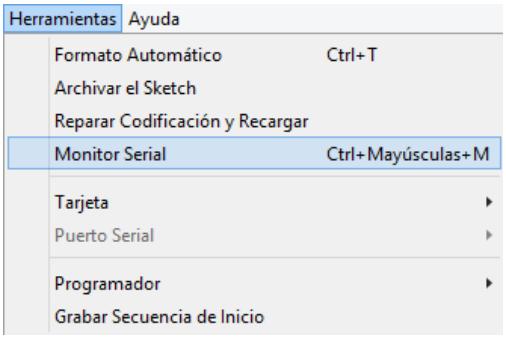

Figura 5. Abrir Monitor Serial

Al abrir el Monitor Serial la placa envía datos vía al monitor tal como muestra la

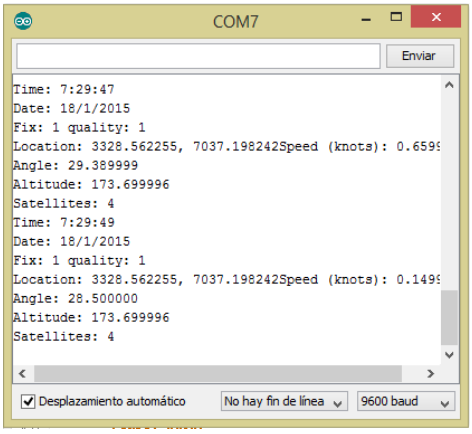

Figura 6. Código ejecutándose

Aplicación de prueba

Como ya se ha mencionado esta aplicación ha sido creada con la finalidad de realizar pruebas de comunicación entre la tarjeta y un dispositivo móvil con sistema Android.

En lo que sigue se detallan los pasos a seguir para instalar la aplicación en un dispositivo móvil asumiendo que el ambiente de programación ya se encuentra configurado. En caso contrario visitar el siguiente link.

Dentro del ambiente Eclipse se debe importar el proyecto haciendo click en *File -> Import…*

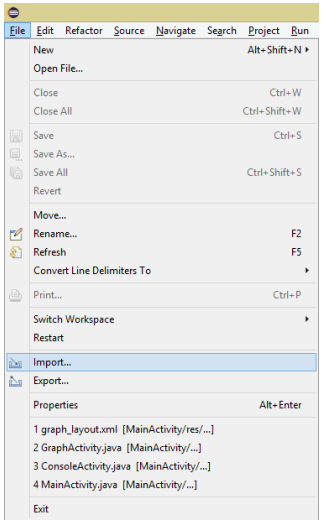

Figura 7. Importar aplicación

Al presionar *Import…* se abrirá la siguiente ventana donde en la sección Android se debe elegir *Existing Android Code Into Workspace*.

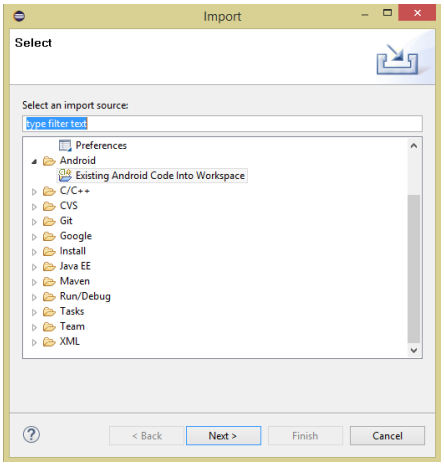

Figura 8. Tipo de importación

Luego se presiona *Next >* para continuar y se visualizará la siguiente ventana.

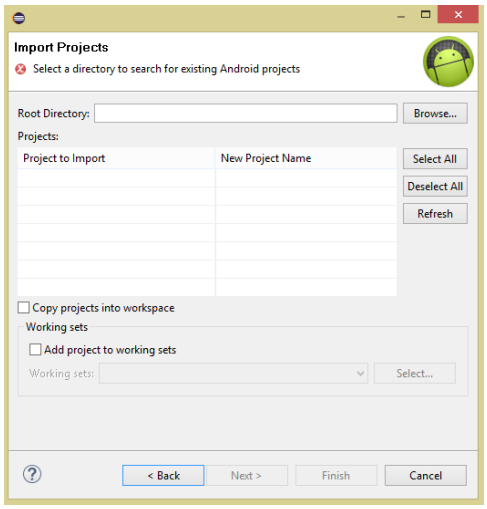

Figura 9. Buscar proyecto

Acá se debe presionar el botón *Browse…* y se visualizará lo siguiente.

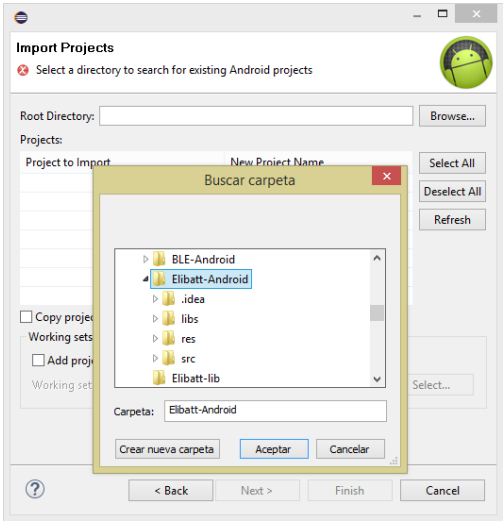

Figura 10. Seleccionar carpeta

Una vez encontrada la ruta donde hemos guardado la carpeta que contiene la aplicación, se presiona *Aceptar*.

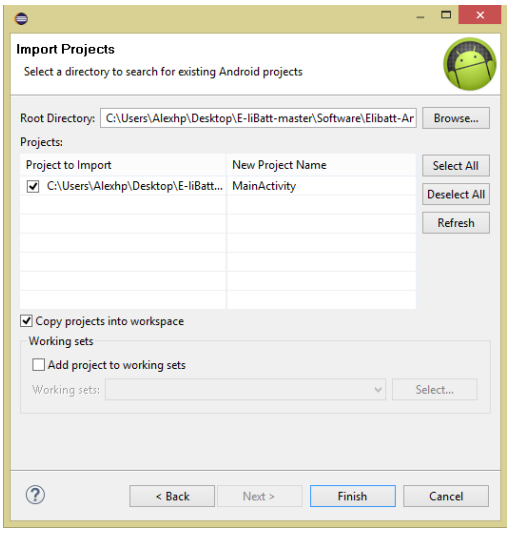

Figura 11. Importar proyecto

Tal como se visualiza en la **Figura 11** se debe verificar que los checkbox's se encuentren de esa forma.

Una vez importada la aplicación, el *Workspace* debe verse de la siguiente forma.

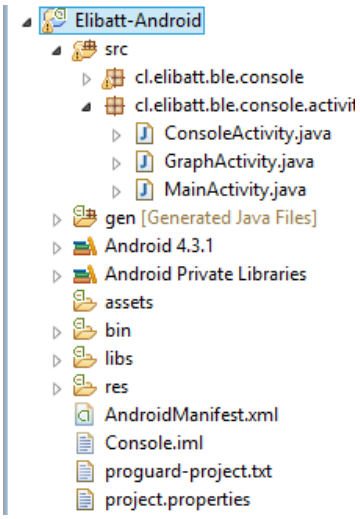

Figura 12. Workspace

Para ejecutar la aplicación, debe encontrarse conectado un dispositivo móvil compatible y presionar sobre el proyecto *Elibatt-Android* con el botón derecho tal como indica la **Figura 13**.

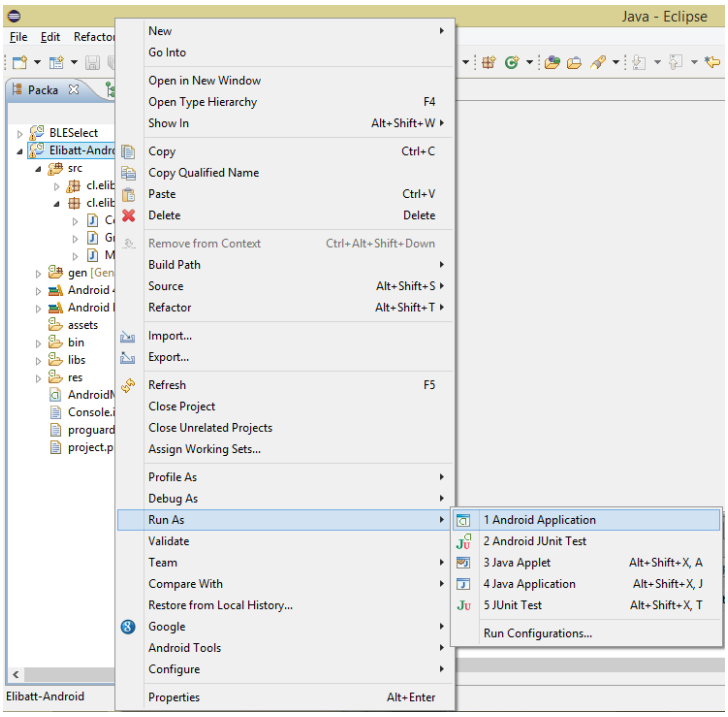

Figura 13. Ejecutar aplicación en dispositivo móvil

Run as -> Android Application permitirá instalar la aplicación en el dispositivo la cual tendrá la siguiente apariencia.

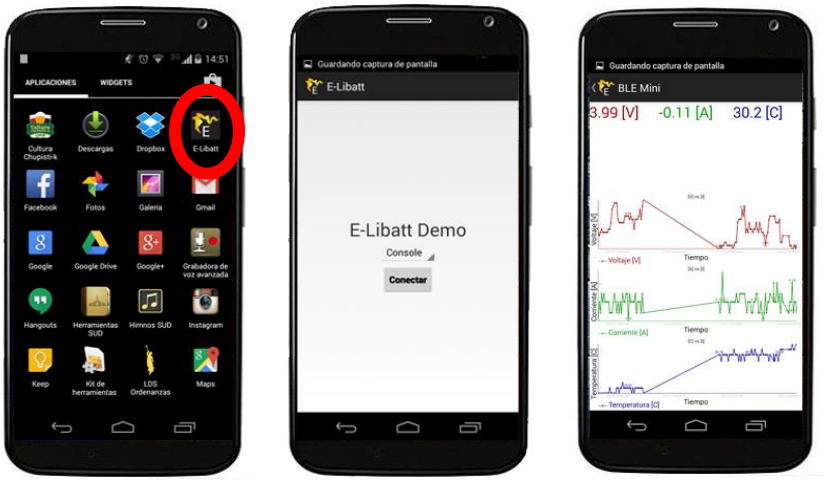

Figura 14. Aplicación ejecutándose

Parte 2: Firmware

El código está compuesto por varias funciones que permiten entregar el funcionamiento adecuado al prototipo. Se trabaja mediante comandos, mostrados en la Tabla 1, que buscan solicitar a la placa información de interés.

Tabla 1. Lista de comandos

Comando	Descripción
T	Entrega la temperatura actual
V	Entrega el voltaje actual
C	Entrega la corriente actual
GPS	Entrega información del módulo de GPS. Fix, latitud, longitud, altura en formato JSON
LA	Entrega la latitud actual
LO	Entrega la longitud actual
H	Entrega la altura o altitud actual
ls	Entrega una lista de todos los archivos de la carpeta root de la SD card.
mv <filename1> <filename2>	Cambia el nombre del archivo <filename1> por el del archivo <filename2>.
rm <filename>	Elimina el archivo llamado <filename>.
cat <filename>	Imprime el contenido del archivo llamado <filename>.
touch <filename>	Crea un archivo vacío llamado <filename>.

Se utilizan las siguientes librerías mostradas en la Tabla 2.

Tabla 2. Librerías utilizadas

Módulo	Librería
Sensor de Temperatura	Wire.h
	Adafruit_MCP9808.h
BLE mini	Arduino.h
	AltSoftSerial.h
Sensor de Corriente y Voltaje	AnalogInput.h
GPS	Adafruit_GPS.h
	ArduinoSoftwareSerial.h
	Endpoint.h
	GPSInput.h
	Utils.h
Lector de Tarjetas SD	SD.h

Al tratarse de una placa Arduino, el código consta de dos funciones principales llamadas *Setup()* y *Loop()*.

Setup

Esta función es llamada al comienzo del código y se ejecuta solo una vez. No posee entradas ni salidas. Se encarga de inicializar la comunicación serial entre el prototipo y el dispositivo móvil. También inicializa el sensor de temperatura y la tarjeta SD. Por último, configura las entradas análogas para los sensores de corriente y voltaje.

Loop

Esta función es llamada luego de ejecutar *Setup()* y realiza justamente lo que su nombre indica. Genera bucles de forma consecutiva permitiendo al código poder realizar acciones de control en forma activa.

Acá se reciben los datos enviados desde un dispositivo móvil conectado vía Bluetooh y se verifica si el carácter enviado corresponde a un comando correcto mediante la función *parseCmd*.

Luego hay una etapa de manejo de sensores donde se recopilan los datos muestreados y se envían vía serial tanto al Monitor Serial como a la aplicación Android vía Bluetooh.

Además, cuenta con un manejo del módulo de GPS donde se obtienen datos como latitud, longitud y altura, también tiempo y fecha.

Por último, una vez recopilados los datos de los sensores y del GPS, se procede a almacenar la información en la tarjeta SD mediante una cadena de texto, donde cada valor va separado de una coma (voltaje, corriente, temperatura, latitud, longitud, altura).

parseCmd

Entradas:

char *s : Cadena de texto, con un comando (string que termina en \n o de largo <= BUFFER_MAX)

int len : Número de caracteres sin contar el \n o BUFFER_MAX.

char *f1 : Posición de memoria donde el método pondrá el nombre del primer archivo

char *f2 : Posición de memoria donde el método pondrá el nombre del segundo archivo

Salidas:

Int : Retorna un entero entre 0 y 11 dependiendo del comando utilizado.

Para explicar el funcionamiento de este método se introduce el concepto de token. Un token es una palabra en una cadena de texto, por ejemplo, "a a a a" tiene 4 tokens. "aa a a" tiene 3 tokens. " a " tiene 1 solo token.

Lo que hace el método parseCmd es lo siguiente:

1) Consume los primeros caracteres de s, y mueve el puntero; por ejemplo, si los primeros caracteres son ls, rm, mv, entonces los reconoce y mueve el inicio de s en 3 posiciones para que calce con el primer argumento del comando. Al extraer el primer token borra los espacios al principio de la cadena, por lo tanto es lo mismo "mv 1" que "mv 1".

2) Para cada comando, la sintaxis es diferente, así que hay un if por cada comando.

3) Luego verifica que el método efectivamente leyó un token, en caso contrario *readNextToken* retorna -1. Finalmente coloca un '\0' al final de f1 para que sea un C string válido (último carácter nulo).

4) Para mv, lo que sigue es mover el puntero s, skipped caracteres más allá (skipped es el largo de f1, que fue "extraído" de s) y acortar el largo de s en skipped caracteres (len -= skipped).

5) Lo que sigue es leer el siguiente token (el segundo nombre de archivo que necesita el comando mv).

6) Por último, si el parseo de s es correcto se retorna de 0 a 11 o se retorna -1 si hay error.

startsWith

Entradas:

char *s : Cadena de texto, con el comando enviado desde el dispositivo.

int len_s : Número de caracteres sin contar el \n o BUFFER_MAX.

char *t : Cadena de texto, con el comando a analizar.

int len_t : Número de caracteres del comando a analizar.

Salidas:

bool : Retorna true si s y t comienzan con el mismo token false en caso contrario.

Este método verifica si las cadenas de texto s y t comienzan con el mismo token. Para ello recurre al método *equals*.

equals

Entradas:

char *s : Cadena de texto, con el comando enviado desde el dispositivo.

int len_s : Número de caracteres del comando a analizar.

char *t : Cadena de texto, con el comando a analizar.

int len_t : Número de caracteres del comando a analizar.

Salidas:

bool : Retorna true si s y t son idénticos, false en caso contrario.

Este método recorre completamente las cadenas de texto y verificar caracter por caracter si es que estos son iguales. Retorna true si son idénticos y false en el caso de que sean distintos.

ReadNextToken

Entradas:

char *s : Una cadena de texto, por ejemplo, con un comando "mv file1" (en cuyo caso extraería el primer token "mv").

int len : Número de caracteres de la cadena s.

char *t : Dirección de memoria donde irá el resultado extraído (en el caso anterior, apuntaría a una zona con el valor "mv").

int &r : Posición en s, donde el ultimo caracter de t fue encontrado (en el caso anterior, sería 1; el último caracter de t, que es v, está en la posición 1 de s).

int max_t_len : Máximo número de caracteres del resultado t (en el código uso FILENAME_MAX, porque estoy extrayendo como tokens nombres de archivo)

Salidas:

Int i

La lógica de esta función se basa en un token. En general, ese método lo que hace es leer conjuntos consecutivos de non-space characters y retorna el número de caracteres que hay que saltarse al extraer el primer token de la cadena.

Temperature

Este método se encarga de obtener el valor actual del sensor de temperatura tanto en grados Celcius como en Fahrenheit y envía el valor en grados Celcius al dispositivo móvil vía Bluetooh.

Current

Este método se encarga de obtener el valor actual del sensor de corriente y envía el valor en Amperes al dispositivo móvil vía Bluetooh. La lectura del valor entregado por el sensor es convertido a voltaje mediante la función *toVolts.* Este voltaje calculado es procesado mediante la función *toAmps* que utiliza una formula dada por el fabricante para entregar finalmente el valor de la corriente.

Voltaje

Este método se encarga de obtener el valor actual del sensor de voltaje y envía el valor en Volt al dispositivo móvil vía Bluetooh. La lectura del valor entregado por el sensor es convertido a voltaje mediante la función *toVolts.* Este voltaje calculado se multiplica por unas resistencias y se obtiene el valor final.

GPS

Este método se encarga de obtener los datos actuales entregados por el GPS y envía estos datos al dispositivo móvil vía Bluetooh en formato JSON. El primer campo corresponde al Fix, el cual indica si el módulo ha realizado una conexión con algún satélite. Los campos restantes entregan latitud, longitud y altura.

Latitud

Este método se encarga de obtener la latitud actual proporcionada por el GPS y envía su valor al dispositivo móvil vía Bluetooh.

Longitud

Este método se encarga de obtener la longitud actual proporcionada por el GPS y envía su valor al dispositivo móvil vía Bluetooh.

Altura

Este método se encarga de obtener la altura actual proporcionada por el GPS y envía su valor al dispositivo móvil vía Bluetooh.

ls

Entrega una lista de todos los archivos de la carpeta root de la SD card mediante el uso de la librería SD.

mv

Entradas:

char *filename1 : Nombre del archivo el cual será modificado con el nombre del segundo argumento.

char *filename2 : Nombre del archivo del cual se obtendrá el nombre para reemplazar en el primer argumento.

El método mv cambia el nombre del archivo <filename1> por el del archivo <filename2> mediante el uso de la librería SD.

rm

Entradas:

char *filename : Nombre del archivo el cual será eliminado.

Elimina el archivo llamado <filename> mediante el uso de la librería SD.

cat

Entradas:

char *filename : Nombre del archivo del cual se imprime su contenido.

Imprime el contenido del archivo llamado <filename> mediante el uso de la librería SD.

touch

Entradas:

char *filename : Nombre del archivo nuevo.

Crea un archivo vacío llamado <filename> mediante el uso de la librería SD.