info@dune-sistemi.com

www.dune-sistemi.com

00183 Roma – via Tracia, 4

Tel: +06-70451252 Fax: +06-77200919

00183 Roma – via Britannia, 54

Tel: +06-77203350 Fax: +06-97605807

AriannaDLLs
API description

Codice:	D0270909
Rev.	1
File:	D0xxxxxx- NT - Arianna BT management.docx
Data:	18/04/2024

	Responsabile	Funzione	Firma	Data
Compilato/Verif.	f. Pucci			18/04/2024
Approvato	F. Andreucci	PJM		18/04/2024
Emesso	f. Pucci	RD		18/04/2024

REVISIONI

Rev.	Data	Autore/i - firma/e	Descrizione
1	18/04/2024	f.Pucci	Prima emissione del documento

Commenti e osservazioni al documento 0270909 (da inviare a Ing. F. Andreucci – tel. 06-77203350 – e-mail: andreucci@dune-sistemi.com)

Segnalazione inviata da: ________________________ data: __________________

Sommario

1. CONTENTS

Arianna BT management and AriannaDLL API description.

This document describes how to use the BT communication with the Arianna system and how to use the AriannaDllV2.dll to get the corrected path.

It lists the command on the BTLE connection and all the API interfaces and how to use them.

2. DESCRIPTION

In the following the escription of the Arianna Device BTLE interface and the AriannaDLLV2.dll API to elaborare the corrected track from the sensor data output.

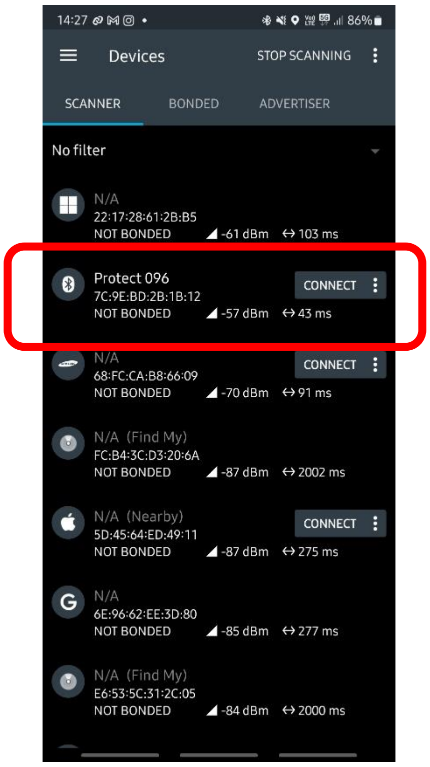
Pay Attention
The Arianna algoritm is implemented inside the Arianna smart sensor, but in order to obtain the reconstructed track (in meters or in Latitude/Longitude) from the sensor output, the AdiannaDLLV2 dll must be used.

2.1 Arianna BTLE management

The arianna BTLE interface on the BTLE is a character based interface, All the data output are strings, all the command are strings.

2.1.1 Arianna Services and Characteristics

In order to manage the Arianna device, you have to connect to the Protect ### device where ### is the number of the device.
In this example ### is 096



14:27 86%

Devices STOP SCANNING

SCANNER BONDED ADVERTISER

No filter

N/A
22:17:28:61:2B:B5
NOT BONDED -61 dBm ↔ 103 ms

Protect 096 CONNECT
7C:9E:BD:2B:1B:12
NOT BONDED -57 dBm ↔ 43 ms

N/A CONNECT
68:FC:CA:B8:66:09
NOT BONDED -70 dBm ↔ 91 ms

N/A (Find My)
FC:B4:3C:D3:20:6A
NOT BONDED -87 dBm ↔ 2002 ms

N/A (Nearby) CONNECT
5D:45:64:ED:49:11
NOT BONDED -87 dBm ↔ 275 ms

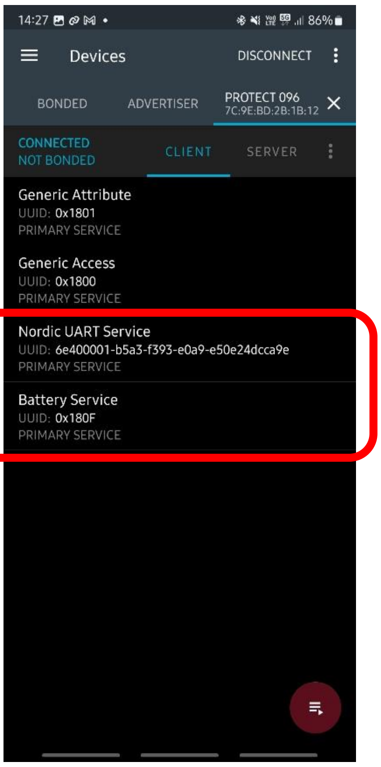
N/A
6E:96:62:EE:3D:80
NOT BONDED -85 dBm ↔ 277 ms

N/A (Find My)
E6:53:5C:31:2C:05
NOT BONDED -84 dBm ↔ 2000 ms

Figure 1 Arianna device services

The Arianna Device export two services:

- The battery service 0x180F
- The Nordic UART service



Generic Attribute
UUID: 0x1801
PRIMARY SERVICE

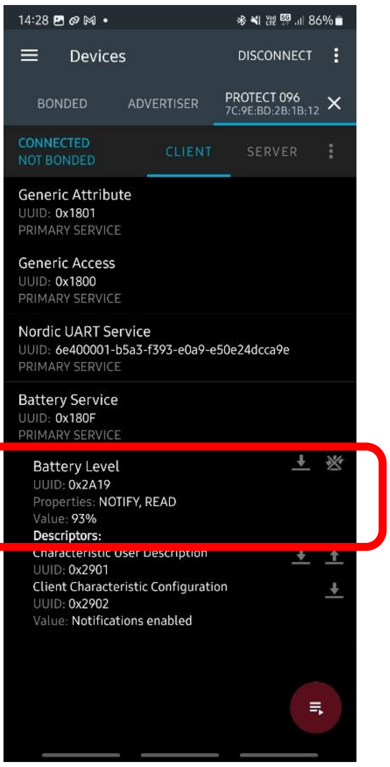
Generic Access
UUID: 0x1800
PRIMARY SERVICE

Nordic UART Service
UUID: 6e400001-b5a3-f393-e0a9-e50e24dcca9e
PRIMARY SERVICE

Battery Service
UUID: 0x180F
PRIMARY SERVICE

Figure 2 Arianna device Nordic UART service and the standard battery service.

The Battery service notify the Arianna device battery level (in %)


14:28 86%

Devices DISCONNECT

BONDED ADVERTISER PROTECT 096 7C:9E:BD:2B:1B:12

CONNECTED
NOT BONDED

CLIENT SERVER

Generic Attribute
UUID: 0x1801
PRIMARY SERVICE

Generic Access
UUID: 0x1800
PRIMARY SERVICE

Nordic UART Service
UUID: 6e400001-b5a3-f393-e0a9-e50e24dcca9e
PRIMARY SERVICE

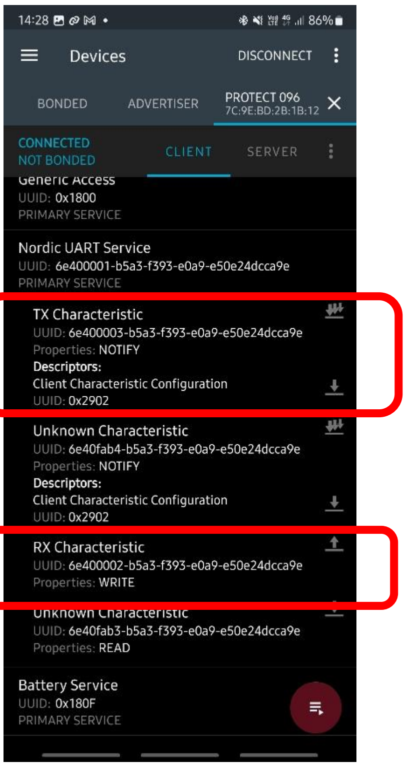
Battery Service
UUID: 0x180F
PRIMARY SERVICE

Battery Level
UUID: 0x2A19
Properties: NOTIFY, READ
Value: 93%
Descriptors:
Characteristic User Description
UUID: 0x2901
Client Characteristic Configuration
UUID: 0x2902
Value: Notifications enabled

Figure 3 Ariana device output for notify.

The Nordic UART service (**6e400001**-b5a3-f393-e0a9-e50e24dcca9e) export the following characteristics:

- **6e400002**-b5a3-f393-e0a9-e50e24dcca9e (Write)
- **6e400003**-b5a3-f393-e0a9-e50e24dcca9e (Notify)



Generic Access
UUID: 0x1800
PRIMARY SERVICE

Nordic UART Service
UUID: 6e400001-b5a3-f393-e0a9-e50e24dcca9e
PRIMARY SERVICE

TX Characteristic
UUID: 6e400003-b5a3-f393-e0a9-e50e24dcca9e
Properties: NOTIFY
Descriptors:
Client Characteristic Configuration
UUID: 0x2902

Unknown Characteristic
UUID: 6e40fab4-b5a3-f393-e0a9-e50e24dcca9e
Properties: NOTIFY
Descriptors:
Client Characteristic Configuration
UUID: 0x2902

RX Characteristic
UUID: 6e400002-b5a3-f393-e0a9-e50e24dcca9e
Properties: WRITE

Unknown Characteristic
UUID: 6e40fab3-b5a3-f393-e0a9-e50e24dcca9e
Properties: READ

Battery Service
UUID: 0x180F
PRIMARY SERVICE

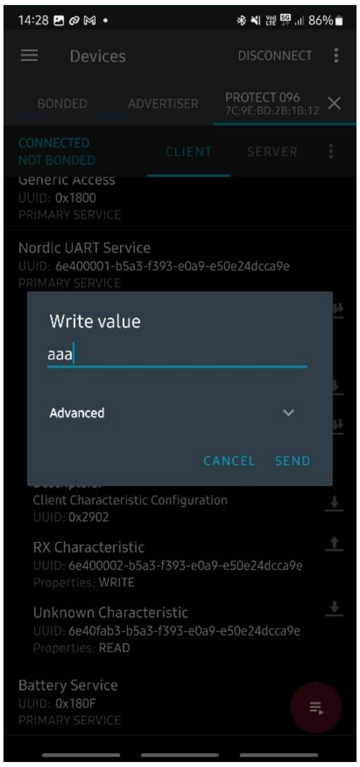
Figure 4 Arianna device Write and Notify characteristics over the Nordic UART service..

The command can be sent on the Write characteristics.

This is a list of some of the implemented command:

• GetServerInfo	get the setup information for the WiFi connection
• DisableWiFi	disable the WiFi connection
• EnableWiFi	Enable the WiFi connection
• SetAriannaServerAddr	Set the Arianna TcpIp server address
• Set AriannaServerPort	Set the Arianna TcpIp server port
• SetSSID	Set the WiFi SSID name
• SetPass	Set the WiFi Password
• aaa	start the mission.

To send a command, a string must be sent on the related characteristics.


14:28 86%

Devices DISCONNECT

BONDED ADVERTISER PROTECT 096 7C:9E:BD:2B:1B:12

CONNECTED NOT BONDED CLIENT SERVER

Generic Access
UUID: 0x1800
PRIMARY SERVICE

Nordic UART Service
UUID: 6e400001-b5a3-f393-e0a9-e50e24dcca9e
PRIMARY SERVICE

Write value
aaa

Advanced

CANCEL SEND

Client Characteristic Configuration
UUID: 0x2902

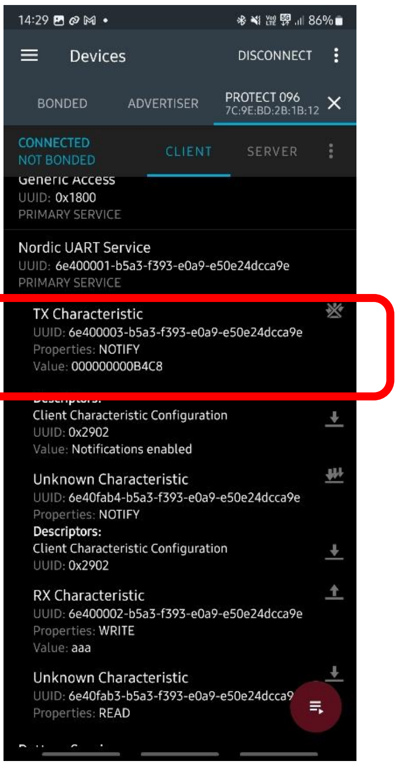
RX Characteristic
UUID: 6e400002-b5a3-f393-e0a9-e50e24dcca9e
Properties: WRITE

Unknown Characteristic
UUID: 6e40fab3-b5a3-f393-e0a9-e50e24dcca9e
Properties: READ

Battery Service
UUID: 0x180F
PRIMARY SERVICE

Figure 5 Start “aaa” command on the write characteristics

On the Notify the data output can be recived.


14:29 86%

Devices DISCONNECT

BONDED ADVERTISER PROTECT 096 7C:9E:BD:2B:1B:12

CONNECTED NOT BONDED CLIENT SERVER

Generic Access
UUID: 0x1800
PRIMARY SERVICE

Nordic UART Service
UUID: 6e400001-b5a3-f393-e0a9-e50e24dcca9e
PRIMARY SERVICE

TX Characteristic
UUID: 6e400003-b5a3-f393-e0a9-e50e24dcca9e
Properties: NOTIFY
Value: 000000000B4C8

Client Characteristic Configuration
UUID: 0x2902
Value: Notifications enabled

Unknown Characteristic
UUID: 6e40fab4-b5a3-f393-e0a9-e50e24dcca9e
Properties: NOTIFY
Descriptors:
Client Characteristic Configuration
UUID: 0x2902

RX Characteristic
UUID: 6e400002-b5a3-f393-e0a9-e50e24dcca9e
Properties: WRITE
Value: aaa

Unknown Characteristic
UUID: 6e40fab3-b5a3-f393-e0a9-e50e24dcca9
Properties: READ

Figure 6 Notify characteristic.

Pay Attention

Bacause the standard BTLE payload is limited to 20 byte, any string longer of 20 bytes is splitted into several string of at least 20 bytes.

2.1.2 Sequence of operations

The arianna device can works on three different communication channels:

- RS232 (UART) on a dedicated connector
- BTLE
- WiFi

In order to select BTLE or WiFI the command EnableWiFi or DisableWiFi must be sent.

If the device is connected on the WiFi the command can be sent over the TcpIp communication or if the WiFi hot spot connection is not available, the device switchs back to the BTLE connection in 30 seconds.

So if the device is configured to use the WiFi and you need to switch back the device to use BTLE:

- disable the WiFi hot spot,
- wait 30 seconds,
- connect on BTLE,
- send the comand DisableWiFI,
- It wil reboot into the permanent BTLE mode.

If the device is already configured in BTLE, connect to the device and send the command **aaa**.

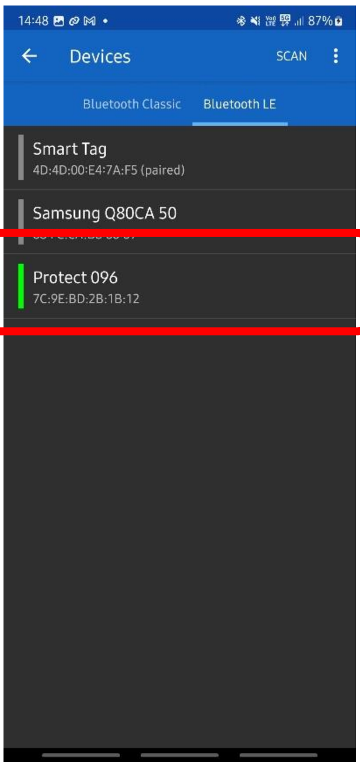
This command lets the device start through the following steps:

- Warm Up mode. It warms up the device for at least 30 seconds.
- Stay Still mobe. Initial setup, it waits for a steady still phase for at least 3 seconds.
- Walking mode. This is the normal mode operation.

Pay attention

- The warm up mode can be transparent for the user. Power on the device as soon as possible and let it warm up.
- The Stay Still mode is crucial for the mission and MUST be done after the aaa command. In this phase the user has to let the sensor (on the foot) stil for at least 3 seconds. If this phase is correctly completed the output messages change accordingly.
- The Walking mode is the normal mode in which the user can walk freely in any direction.

Look at the string output to check the status of the sensor.

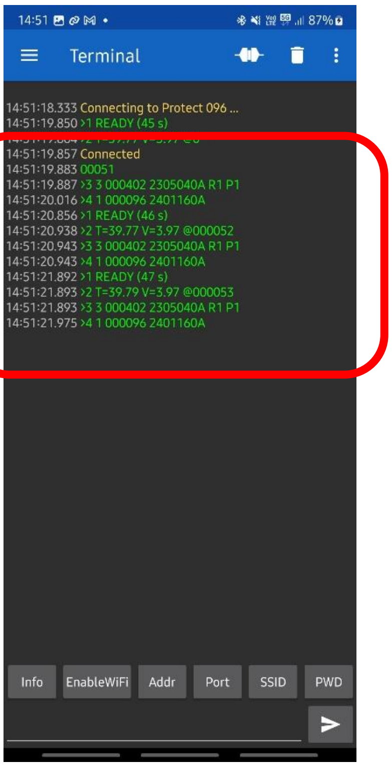


Smart Tag
4D:4D:00:E4:7A:F5 (paired)

Samsung Q80CA 50

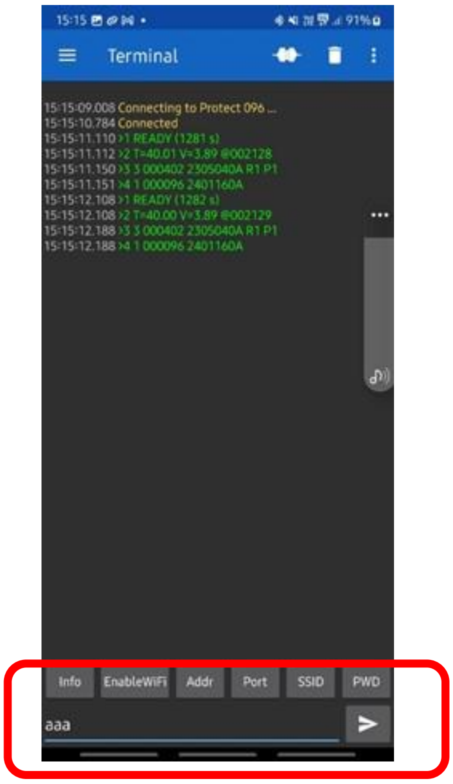
Protect 096
7C:9E:BD:2B:1B:12

Figure 7 Connect to the Protect 096 Arianna device.



```
14:51:18.333 Connecting to Protect 096 ...
14:51:19.850 >1 READY (45 s)
14:51:19.864 >2 T=39.77 V=3.97 @0
14:51:19.857 Connected
14:51:19.883 00051
14:51:19.887 >3 3 000402 2305040A R1 P1
14:51:20.016 >4 1 000096 2401160A
14:51:20.856 >1 READY (46 s)
14:51:20.938 >2 T=39.77 V=3.97 @000052
14:51:20.943 >3 3 000402 2305040A R1 P1
14:51:20.943 >4 1 000096 2401160A
14:51:21.892 >1 READY (47 s)
14:51:21.893 >2 T=39.79 V=3.97 @000053
14:51:21.893 >3 3 000402 2305040A R1 P1
14:51:21.975 >4 1 000096 2401160A
```

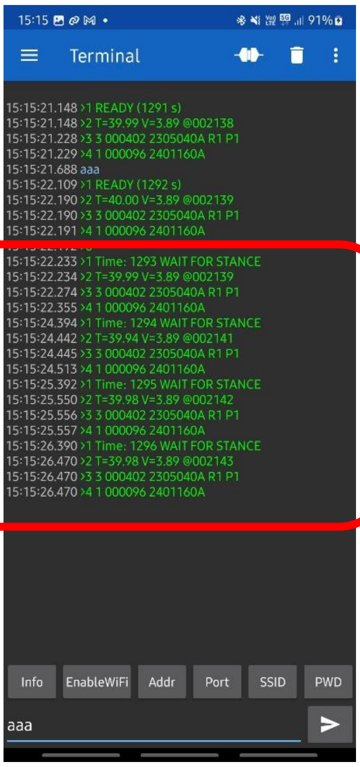
*Figure 8 Standard output after the connection (**READY**) and befor the aaa command.*



```
15:15:09.008 Connecting to Protect 096 ...
15:15:10.784 Connected
15:15:11.110 >1 READY (1281 s)
15:15:11.112 >2 T=40.01 V=3.89 @002128
15:15:11.150 >3 3 000402 2305040A R1 P1
15:15:11.151 >4 1 000096 2401160A
15:15:12.108 >1 READY (1282 s)
15:15:12.108 >2 T=40.00 V=3.89 @002129
15:15:12.188 >3 3 000402 2305040A R1 P1
15:15:12.188 >4 1 000096 2401160A
```

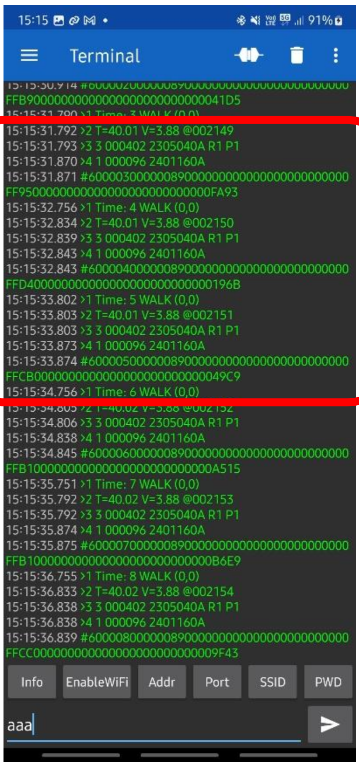
```
aaa
```

Figure 9 send the aaa comand.



```
15:15:21.148 >1 READY (1291 s)
15:15:21.148 >2 T=39.99 V=3.89 @002138
15:15:21.228 >3 3 000402 2305040A R1 P1
15:15:21.229 >4 1 000096 2401160A
15:15:21.688 aaa
15:15:22.109 >1 READY (1292 s)
15:15:22.190 >2 T=40.00 V=3.89 @002139
15:15:22.190 >3 3 000402 2305040A R1 P1
15:15:22.191 >4 1 000096 2401160A
15:15:22.233 >1 Time: 1293 WAIT FOR STANCE
15:15:22.234 >2 T=39.99 V=3.89 @002139
15:15:22.274 >3 3 000402 2305040A R1 P1
15:15:22.355 >4 1 000096 2401160A
15:15:24.394 >1 Time: 1294 WAIT FOR STANCE
15:15:24.442 >2 T=39.94 V=3.89 @002141
15:15:24.445 >3 3 000402 2305040A R1 P1
15:15:24.513 >4 1 000096 2401160A
15:15:25.392 >1 Time: 1295 WAIT FOR STANCE
15:15:25.550 >2 T=39.98 V=3.89 @002142
15:15:25.556 >3 3 000402 2305040A R1 P1
15:15:25.557 >4 1 000096 2401160A
15:15:26.390 >1 Time: 1296 WAIT FOR STANCE
15:15:26.470 >2 T=39.98 V=3.89 @002143
15:15:26.470 >3 3 000402 2305040A R1 P1
15:15:26.470 >4 1 000096 2401160A
```

*Figure 10 Stay Still phase. Look at the WAIT FOR STANCE command. **This phase ends only if the sensor stays still for at least 3 seconds**.*



```
15:15:30.914 #600002000000089000000000000000000000000000
FFB900000000000000000000000000041D5
15:15:31.790 >1 Time: 3 WALK (0,0)
15:15:31.792 >2 T=40.01 V=3.88 @002149
15:15:31.793 >3 3 000402 2305040A R1 P1
15:15:31.870 >4 1 000096 2401160A
15:15:31.871 #600003000000089000000000000000000000000000
FF9500000000000000000000000000FA93
15:15:32.756 >1 Time: 4 WALK (0,0)
15:15:32.834 >2 T=40.01 V=3.88 @002150
15:15:32.839 >3 3 000402 2305040A R1 P1
15:15:32.843 >4 1 000096 2401160A
15:15:32.843 #600004000000089000000000000000000000000000
FFD400000000000000000000000000196B
15:15:33.802 >1 Time: 5 WALK (0,0)
15:15:33.803 >2 T=40.01 V=3.88 @002151
15:15:33.803 >3 3 000402 2305040A R1 P1
15:15:33.873 >4 1 000096 2401160A
15:15:33.874 #600005000000089000000000000000000000000000
FFCB00000000000000000000000000049C9
15:15:34.756 >1 Time: 6 WALK (0,0)
15:15:34.805 >2 T=40.02 V=3.88 @002152
15:15:34.806 >3 3 000402 2305040A R1 P1
15:15:34.838 >4 1 000096 2401160A
15:15:34.845 #600006000000089000000000000000000000000000
FFB10000000000000000000000000A515
15:15:35.751 >1 Time: 7 WALK (0,0)
15:15:35.792 >2 T=40.02 V=3.88 @002153
15:15:35.792 >3 3 000402 2305040A R1 P1
15:15:35.874 >4 1 000096 2401160A
15:15:35.875 #600007000000089000000000000000000000000000
FFB10000000000000000000000000B6E9
15:15:36.755 >1 Time: 8 WALK (0,0)
15:15:36.833 >2 T=40.02 V=3.88 @002154
15:15:36.838 >3 3 000402 2305040A R1 P1
15:15:36.838 >4 1 000096 2401160A
15:15:36.839 #600008000000089000000000000000000000000000
FFCC00000000000000000000000009F43
```

Figure 11Normal working (WALK). The 74h string are the payload for the track reconstruction.

74h string example:

#010017000000B900000000000000000000000000006000000000000000000100000000023C6

Each 74h string must be used to feed the AriannaDllV2 library in order to reconstruct the track path.

2.2 AriannaDLLV2.dll

This DLL implements the management of the 74H strings produced by the sensor and creates and updates the complete track result.
It includes three different kinds of functions:

1) Initialization and closing of the DLL internal structures.
2) management of the 74h strings and function to retrieve the full and corrected track.

3) Auxiliary functions to retrieve some information on the internal status.

2.2.1 DLL functiones

The dll is written in C++ and can be compiled for Windows / Linux, ….

In the follows the DLL function included in the AriannaDLLV2 and the signature to call the DLL function from a C# example.

```
/// <summary>
/// DLL Arianna 2.0
/// </summary>
/// <param name="hdc"></param>
/// <param name="nIndex"></param>
/// <returns></returns>
[DllImport("ariannaDllv2.dll",
      CharSet = CharSet.Ansi,
      CallingConvention = CallingConvention.Cdecl)]
unsafe static extern int getAriannaPostprocVersion();

/// <summary>
/// DLL Arianna 2.0 Inizializzazione libreria
/// </summary>
/// <param name="none"></param>
/// <returns></returns>
[DllImport("ariannaDllv2.dll",
CharSet = CharSet.Ansi,
CallingConvention = CallingConvention.Cdecl)]
unsafe static extern int initAriannaLib();

/// <summary>
/// DLL Arianna 2.0 init pat operatore opo
/// </summary>
/// <param name="id"></param>
/// <param name="tipo"></param> modalità 74h / 98h
/// <returns></returns>
[DllImport("ariannaDllv2.dll",
      CharSet = CharSet.Ansi,
      CallingConvention = CallingConvention.Cdecl)]
unsafe static extern int initAriannaPath(int op,int winStepNum98h,int calPeriod98h);

///-----------------------------------------------------------------------

/// <summary>
/// DLL Arianna 2.0 gestione stringha 74h
/// </summary>
/// <param name="op"></param>
/// <param name="str74"></param>
/// <returns></returns>
[DllImport("ariannaDllv2.dll",
      CharSet = CharSet.Ansi,
      CallingConvention = CallingConvention.Cdecl)]
unsafe static extern int setAriannaMessage(int op, char[] str74h_98h);

/// <summary>
/// DLL Arianna 2.0
/// </summary>
/// <param name="op"></param>
/// <param name="step"></param>
/// <param name="field"></param>
/// <returns></returns>
[DllImport("ariannaDllv2.dll",
      CharSet = CharSet.Ansi,
      CallingConvention = CallingConvention.Cdecl)]
unsafe static extern float getAriannaData(int op, int step, int field);

/// <summary>
/// DLL Arianna 2.0
/// </summary>
/// <param name="op"></param>
/// <param name="x0"></param>
/// <param name="y0"></param>
/// <param name="rotation"></param>
/// <param name="deskewing"></param>
/// <param name="type"></param>
/// <returns></returns>
[DllImport("ariannaDllv2.dll",
      CharSet = CharSet.Ansi,
      CallingConvention = CallingConvention.Cdecl)]
unsafe static extern IntPtr getAriannaPath(int op, float X0, float Y0, float rotation,
float deskewing, int type);

/// <summary>
/// DLL Arianna 2.0
/// </summary>
/// <param name="op"></param>
/// <param name="x0"></param>
/// <param name="y0"></param>
/// <param name="rotation"></param>
/// <param name="deskewingT"></param>
/// <param name="deskewingS"></param>
/// <param name="type"></param>
/// <param name="latitude"></param>
/// <param name="longitude"></param>
/// <returns></returns>
[DllImport("ariannaDllv2.dll",
      CharSet = CharSet.Ansi,
      CallingConvention = CallingConvention.Cdecl)]
unsafe static extern IntPtr getAriannaPath_GPS(int op, float X0, float Y0, float
rotation, float deskewingT, float deskewingP, int type, double Lat0, double Lon0);

/// <summary>
/// DLL Arianna 2.0
/// </summary>
/// <param name="op"></param>
/// <param name="field"></param>
/// <returns></returns>
[DllImport("ariannaDllv2.dll",
      CharSet = CharSet.Ansi,
      CallingConvention = CallingConvention.Cdecl)]
unsafe static extern IntPtr getAriannaElevation(int op, int field);

///------------------------------------------------------------------------
/// <summary>
/// DLL Arianna 2.0
/// </summary>
/// <param name="op"></param>
/// <returns></returns>
```

```
[DllImport("ariannaDllv2.dll",
      CharSet = CharSet.Ansi,
      CallingConvention = CallingConvention.Cdecl)]
unsafe static extern int getAriannaStep(int op);
/// <summary>
/// DLL Arianna 2.0
/// </summary>
/// <param name="op"></param>
/// <returns></returns>
[DllImport("ariannaDllv2.dll",
      CharSet = CharSet.Ansi,
      CallingConvention = CallingConvention.Cdecl)]
unsafe static extern int getAriannaMsg(int op);

/// <summary>
/// DLL Arianna 2.0
/// </summary>
/// <param name="op"></param>
/// <returns></returns>
[DllImport("ariannaDllv2.dll",
      CharSet = CharSet.Ansi,
      CallingConvention = CallingConvention.Cdecl)]
unsafe static extern int getAriannanPassi(int op);
/// <summary>
/// DLL Arianna 2.0
/// </summary>
/// <param name="op"></param>
/// <returns></returns>
[DllImport("ariannaDllv2.dll",
      CharSet = CharSet.Ansi,
      CallingConvention = CallingConvention.Cdecl)]
unsafe static extern int getAriannanMessaggi(int op);

/// <summary>
/// DLL Arianna 2.0
/// </summary>
/// <param name="op"></param>
/// <returns></returns>
[DllImport("ariannaDllv2.dll",
      CharSet = CharSet.Ansi,
      CallingConvention = CallingConvention.Cdecl)]
unsafe static extern int getAriannaGround(int op);

/// <summary>
/// DLL Arianna 2.0
/// </summary>
/// <param name="op"></param>
/// <returns></returns>
[DllImport("ariannaDllv2.dll",
      CharSet = CharSet.Ansi,
      CallingConvention = CallingConvention.Cdecl)]
unsafe static extern float getAriannaBattery(int op);
iannaElevation(int op, int field);

/// <summary>
/// DLL Arianna 2.0
/// </summary>
/// <param name="op"></param>
/// <param name="tipo"></param>
/// <returns></returns>
[DllImport("ariannaDllv2.dll",
      CharSet = CharSet.Ansi,
      CallingConvention = CallingConvention.Cdecl)]
unsafe static extern int freePath(int op);

/// <summary>
/// DLL Arianna 2.0
/// </summary>
/// <returns></returns>
[DllImport("ariannaDllv2.dll",
CharSet = CharSet.Ansi,
CallingConvention = CallingConvention.Cdecl)]
unsafe static extern int freeAriannaAllPath();
```

2.2.1.1 int getAriannaPostprocVersion();

This function return and integer (32bits) that contains the DLL version. To better interpreter this value read is in hexadecimal format:

For example:

ver = 604244234 -> 0x2404090A -> 24/04/09/A

2.2.1.2 int initAriannaLib();

This function must be called one time at the beginning al the DLL use.

2.2.1.3 int freePath(int op);

This function releases all the structures allocated for the operator Op.

Parameters:

Op	operator Id [0:255]

2.2.1.4 int freeAriannaAllPath();

This function releases all the structures allocated for all the operators.

2.2.1.5 int initAriannaPath(int op,int winStepNum98h,int calPeriod98h);

This function must be called at the firs message of each user stream. Op is the user Id (0:255).

Parameters:

Op	operator Id [0:255]

To get this value from the 74h String get the character 2 & 3

For example:

```
var hex = new string(Msg74h.Skip(n).Take(2).ToArray());
op = Convert.ToUInt16(hex, 16);
```

For the 74h messages the values winStep98h and calPeriod98h must be 0.

2.2.1.6 int setAriannaMessage(int op, char[] str74h_98h);

Fhis function has to be called for any 74h string recived.

Parameters:

Op	operator Id [0:255]
str74h	74h string from the sensor.

2.2.1.7 float getAriannaData(int op, int step, int field);

Legacy function.

2.2.1.8 IntPtr getAriannaPath(int op, float X0, float Y0, float rotation, float deskewing, int type);

This function allows the application to retrive the corrected track.

Parameters:

Op	operator Id [0:255]
X0	offset in meters from the starting point (0,0)
Y0	offset in meters from the starting point (0,0)
rotation	auxiliari rotation in Radiants
deskewing	auxiliari compensation of the estimated deskewing
type	Selection of the estimated path:

1 : Backward corrected track with Magnetic correctection.

2: Best Estimation History.

3: legacy.

4: User corrected track.

5: Pure inertial track, non-magnetic correction.

This function returns a vector of 2*nSteps **float** elements. From 0 to nSteps it is the X values in meters from the starting point (0,0); from nStep to 2*nStep it contains the Y values in meters from the starting point (0,0).

The number of steps can be obtained by using the function:

```
	nStep = getAriannaMsg(op);
```

2.2.1.9 IntPtr getAriannaPath_GPS(int op, float X0, float Y0, float rotation, float deskewingT, float deskewingP, int type, double Lat0, double Lon0);

This function allows the application to retrive the corrected track.

Parameters:

Op	operator Id [0:255]
X0	offset in meters from the starting point (0,0)
Y0	offset in meters from the starting point (0,0)
rotation	auxiliari rotation in Radiants
deskewing	auxiliari compensation of the estimated deskewing
type	Selection of the estimated path: 1 : Backward corrected track with Magnetic correctection. 2: Best Estimation History. 3: legacy. 4: User corrected track. 5: Pure inertial track, non-magnetic correction.
Latitude	Latitude of the starting point
Longitude	Longitude of the starting point

This function returns a vector of **double** of 2*nSteps elements. From 0 to nSteps it is the Latitude values; from nStep to 2*nStep it contains the Longitude values in meters from the starting point (0,0).

The number of steps can be obtained by using the function:

```
	nStep = getAriannaMsg(op);
```

2.2.1.10 IntPtr getAriannaElevation(int op, int field);

This function allows the application to retrive the elevation.

Parameters:

Op	operator Id [0:255]
type	Selection of the estimated path: 9 : Inertial altitude estimation 10: Barometric altitude estimation

This function returns a vector of nSteps **float** elements.

The number of steps can be obtained by using the function:

```
    nStep = getAriannaMsg(op);
```

2.2.1.11 int getAriannaMsg(int op);

This function returns the number of the 74h messages recived.

Parameters:

Op	operator Id [0:255]

2.2.1.12 int getAriannanStep(int op);

Legacy.

Parameters:

Op	operator Id [0:255]

2.2.1.13 int getAriannanPassi(int op);

This function returns the number of steps in the estimated track.

Parameters:

Op	operator Id [0:255]

2.2.1.14 int getAriannanMessaggi(int op);

This function returns the number of steps in the estimated track.

Parameters:

Op	operator Id [0:255]

2.2.1.15 int getAriannaGround(int op);

This function returns the flag that highlights if the user is not standing on foot but it is lying down.

Parameters:

Op	operator Id [0:255]

2.2.1.16 float getAriannaBattery(int op);

This function returns the value of the voltage of the battery.

Parameters:

Op	operator Id [0:255]

2.2.2 Example

```
List<int> ListaOp = new();

var ver            = getAriannaPostprocVersion();
var ret            = initAriannaLib();
var   op           = 0;
var nStep          = 0;
var nMsgx          = 0;
var bGround        = 0;
var nPassi         = 0;
var nMsg           = 0;
var vBatt          = 0.0;

using (var sr = new
       StreamReader("RawData_20240408_090001_000001_000001_001.decod.74h")) {

      int n = 1;
      var table = sr.ReadToEnd();
      var data  = table.Split('\n');
      var lines = data.ToList().FindAll(p => p.Length > 0 && p[0]=='#');

      Stopwatch stopWatch2 = new Stopwatch();
      stopWatch2.Start();

      foreach (var line in lines) {
            if (line.Length > 0) {
                  var hex = new string(line.Skip(n).Take(2).ToArray());
                  op = Convert.ToUInt16(hex, 16);
                  if (!ListaOp.Contains(op)) {
                        ListaOp.Add(op);
                        var reti = initAriannaPath(op, 0, 0);
```

```
                    }
                var err = setAriannaMessage(op, line.ToCharArray());
                if (err >= 0) {
                    nStep  = getAriannaMsg(op);
                    nMsgx  = getAriannaStep         (op);

                    nPassi= getAriannanPassi       (op);
                    nMsg   = getAriannanMessaggi    (op);
                    bGround = getAriannaGround            (op);
                    vBatt  = getAriannaBattery      (op);

                    // Console.WriteLine("Op:" + op + " time:" + nMsg.ToString("N0") +
" nStep:" + nPassi.ToString("N0") + " Battery:" + vBatt.ToString("f2") + "V (" + err +
")\n");

                        float[] resultXm = new float[nStep];
                        float[] resultYm = new float[nStep];
                        double a0 = 0;// (RealTimeRotateAngle - MagneticDeclination)
* UtilityManager.GtoR;
                        var w = 0.0;

                        IntPtr pointerXYc = getAriannaPath_GPS(op,
Convert.ToSingle(0), Convert.ToSingle(0), Convert.ToSingle(a0), Convert.ToSingle(w),
Convert.ToSingle(0), 1, Convert.ToDouble(0), Convert.ToDouble(0));
                        if (pointerXYc == (IntPtr)0)
                            return;
                        float[] resultXYc = new float[2 * nStep];
                        Marshal.Copy(pointerXYc, resultXYc, 0, 2 * nStep);
                        for (int ij = 0; ij < nStep; ij++) {
                            resultXm[ij] = resultXYc[ij];
                            resultYm[ij] = resultXYc[ij + nStep];
                        }

                        IntPtr pointerZp = getAriannaElevation(op, _ZP_VALUE);
                        if (pointerZp == (IntPtr)0)
                            return;
                        float[] resultZp = new float[nStep];
                        Marshal.Copy(pointerZp, resultZp, 0, nStep);

                        IntPtr pointerZi = getAriannaElevation(op, _ZI_VALUE);
                        if (pointerZi == (IntPtr)0)
                            return;
                        float[] resultZi = new float[nStep];
                        Marshal.Copy(pointerZi, resultZi, 0, nStep);

                    } else {
                        Console.WriteLine("Op:" + op + "Error!!!!\n");
                    }
                }
            }

            stopWatch2.Stop();

            TimeSpan ts2 = stopWatch2.Elapsed;

            string elapsedTime2 = String.Format("{0:00}:{1:00}:{2:00}.{3:00}", ts2.Hours,
            ts2.Minutes, ts2.Seconds, ts2.Milliseconds / 10);

            Console.WriteLine("Version " + ver.ToString("X"));
            Console.WriteLine("nStep " + nPassi + "/" + nMsg);
            Console.WriteLine("RunTime " + elapsedTime2);
```

```
}
freePath(op);
```