Application Note

Wireless Stepwise Dead Reckoning

"PDR with MIMU22BL"

Revision 1.1

R&D Centre:
GT Silicon Pvt Ltd
171, MIG, Awadhpuri,
Block B, Lakhanpur,
Kanpur (UP), India, PIN – 208024

Tel: +91 512 258 0039
Fax: +91 512 259 6177
Email: info@inertialelements.com
URL: www.inertialelements.com

Revision History

Revision	Revision Date	Updates
1.0	06 January 2018	Initial version
1.1	23 January 2018	Minor revision & typographical corrections

Purpose & Scope

This document lists down the instructions to construct stepwise dead reckoning (SWDR) data in global (user's) reference frame on an application platform, while using BLE interface for data transmission. It also contains pseudo-code required for this purpose.

Please refer embedded code for better understanding.

Other Reference: MIMU22BL Integration Guide

Wireless Stepwise Dead Reckoning with an Application Platform

How to construct tracked path profile of the wearer of MIMU22BL, when the BLE interface is used for data transmission, is described here.

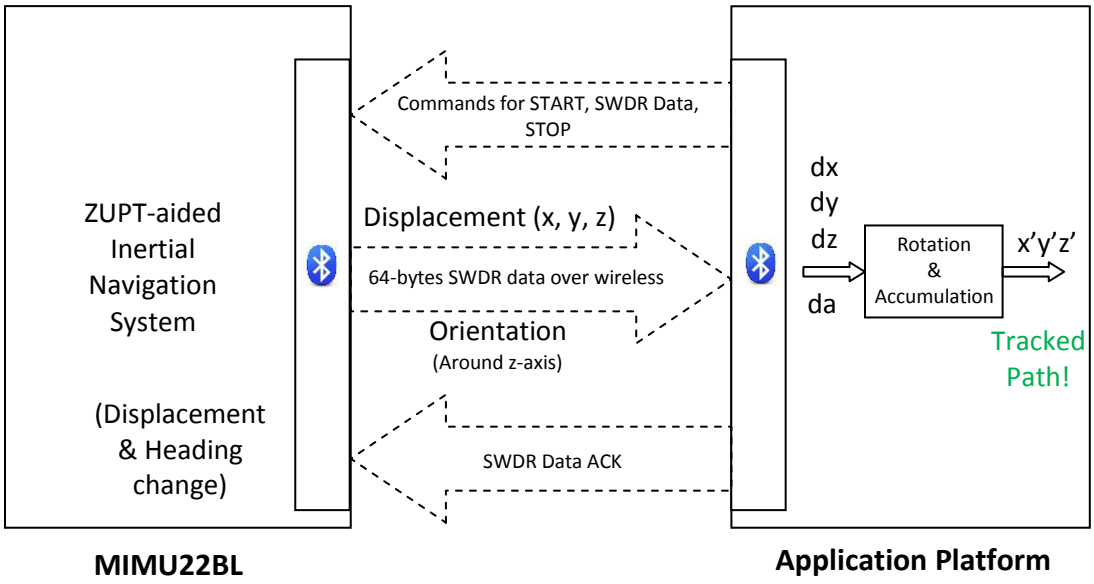


Figure 1: Interfacing of MIMU22BL with an application platform

Application platform can also use the following method to construct path profile with BLE. There is no major difference with the above method except the user can make use of the true compass heading coming out from MIMU22BL.

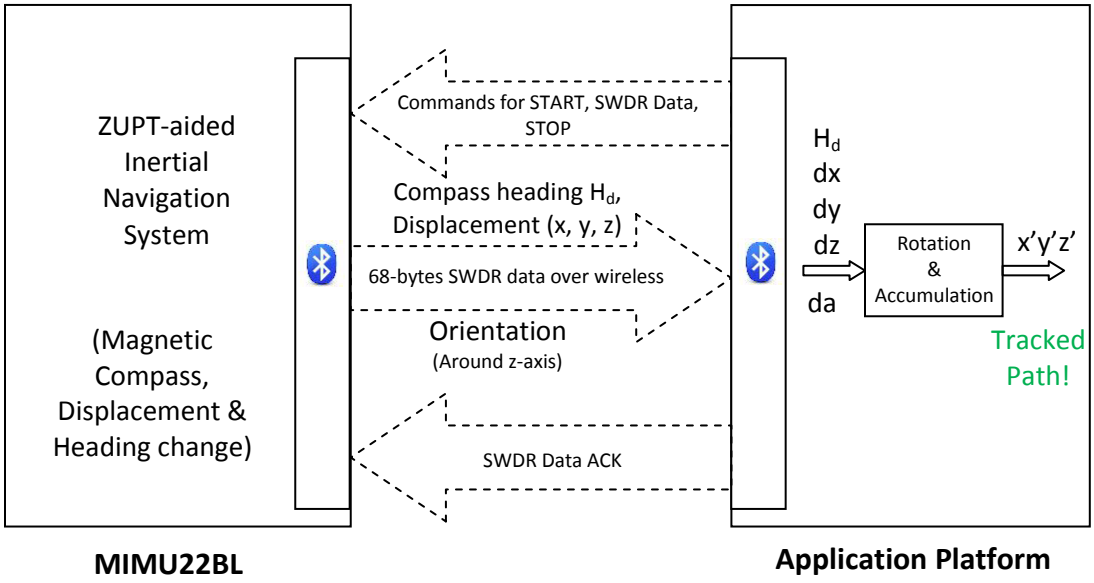


Figure 2: Interfacing of MIMU22BL with application platform to get SWDR data along with true magnetic north.

Command Flow Sequence

Step 1: START - Start receiving data packet from MIMU22BL

Step 2: ACK - Send acknowledgement for last data packet received from MIMU22BL

Step 3: SWDR - MIMU22BL sends DATA packet containing displacement and orientation information for each stride, at every step, to XMIMU22BL. (Step = whenever detects zero motion or standstill is detected).

Step 4: Application platform acknowledges receiving last DATA packet by sending appropriate ACK to MIMU22BL. (Cycle of steps 3 and 4 is repeated until application sends STOP. On receiving STOP command, MIMU22BL executes Step 5)

Step 5: STOP - (i) Stop processing in MIMU22BL (ii) Stop all outputs in MIMU22BL

Description

Step 1: START – Start receiving data packet from MIMU22BL

There are two possible command for SWDR only and SWDR with true compass heading.

a. Command to START transmitting "stepwise dead reckoning"(SWDR) data consists of 3 bytes: {0x34, 0x00, 0x34} .
b. Command to START transmitting "stepwise dead reckoning data with compass heading" data consists of 4 bytes: {0x43, 0x40, 0x00, 0x83}.

START command results in following:

A. MIMU22BL transmits 4 bytes acknowledgement in response to START command from the application platform.

I. For SWDR only

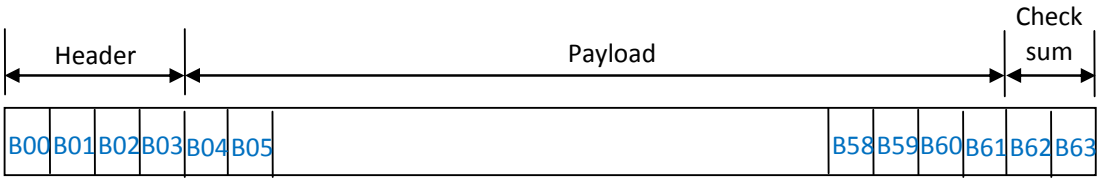
"A0 34 00 D4"
A0: Acknowledgement state
34 = Start command state
00 D4 = Checksum

II. For SWDR with compass heading

"A0 43 00 E3"
A0: Acknowledgement state
43 = Start command state
00 E3 = Checksum

B.

i. Followed by 4-byte acknowledgement, MIMU22BL sends out 64-bytes tracking data in form of packets via wireless communication for SWDR only. Below figure illustrates how the packets look like:



B00: State of the Header
B01-B02: Data packet number
B03: Number of bytes in Payload

B04-B07: dx: Displacement in x (Type float)
B08-B11: dy: Displacement in y (Type float)
B12-B15: dz: Displacement in z (Type float)
B16-B19: da: Change in angle around z axis (Type float)

B20-B59: 10 entries (4 bytes each) of a 4x4 symmetric covariance matrix

B60-B61: Step counter

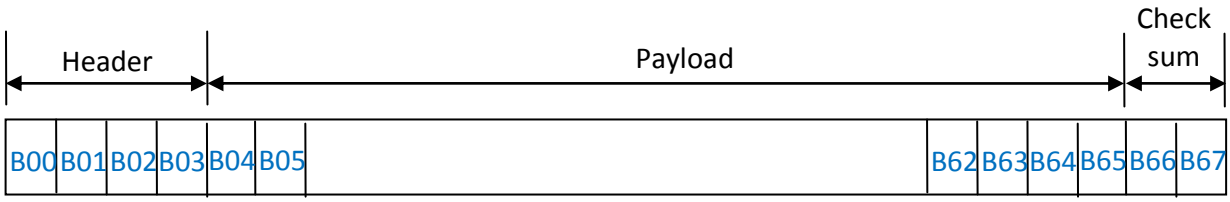
B62-B63: Check sum

Figure 2 Data packet for stepwise dead reckoning

Below is a data packet example:

Figure 3 Total data packet size is 64 bytes. There are 4 header bytes - one for state, two for data packet number and one for Payload size. Number of bytes in Payload is 58. The last 2 bytes are for checksum.

ii. Followed by 4-byte acknowledgement, MIMU22BL sends out 68-bytes tracking data in form of packets via wireless communication for SWDR along with true compass heading. Below figure illustrates how the packets look like:



B00: State of the Header
B01-B02: Data packet number
B03: Number of bytes in Payload
B04-B07:- H_d: Tilt compensated magnetic compass heading
B08-B11: dx: Displacement in x (Type float)
B12-B15: dy: Displacement in y (Type float)
B16-B19: dz: Displacement in z (Type float)
B20-B23: da: Change in angle around z axis (Type float)

B24-B63: 10 entries (4 bytes each) of a 4x4 symmetric covariance matrix

B64-B65: Step counter

B66-B67: Check sum

Figure 4 Data packet for stepwise dead reckoning

Data Packet Description

These packets consist of 64 bytes / 68 bytes based on command.

A. For SWDR only the data format is the followings:

1. **Header**: First 4 bytes are 'Header'.
    a. Among these 4 bytes first is the header which is STATE_OUTPUT_HEADER which is 170 (0xAA).

b. 2nd and 3rd byte jointly tells the data packet number sent by the device since the beginning of stepwise dead reckoning. Thus the packet number is a 16-bits variable.
c. 4th byte depicts the payload i.e. the number of bytes (0x3A) this data packet contains which consists of required information asked by the user.

2. **Payload**: Next is the payload, which consisted of 14 elements of type 'float' thus comprising of 14*4=56 bytes.
   a. First 4 elements consisted of the delta vector i.e. the change in position x, y, z and the angle of rotation around z-axis (the change in the angle of the x-y plane). As these elements are of type float, thus are of 32 bits. The change in position is between two successive ZUPT instances, i.e. the displacement vector from one step to the next one.
   b. The other 10 elements of the payload are used to form the covariance matrix, which is a 4x4 symmetric matrix, thus 10 elements. These are not used in the step-wise dead reckoning. These are used in data fusion from two MIMU22BL devices in case of dual foot mounted system.

3. **Step Counter**: Next two bytes consisted of step counter, which is a counter taking record of ZUPT instances observed. This is essentially number of times velocity of MIMU22BL goes down below predefined threshold. Hence it is the number of steps taken by the wearer, if MIMU22BL is used for foot-mounted pedestrian tracking.

4. **Checksum**: The last two bytes of 64 bytes are consisted of check sum which is sum of all the bytes received prior to these. These are used to cross-check correctness of the data transferred.

B. For SWDR along with compass heading the data format is the followings:

1. **Header**: First 4 bytes are 'Header'.
   a. Among these 4 bytes first is the header which is STATE_OUTPUT_HEADER which is 170 (0xAA).
   b. 2nd and 3rd byte jointly tells the data packet number sent by the device since the beginning of stepwise dead reckoning. Thus the packet number is a 16-bits variable.
   c. 4th byte depicts the payload i.e. the number of bytes (0x3E) this data packet contains which consists of required information asked by the user.

2. **Payload**: Next is the payload, which consisted of 15 elements of type 'float' thus comprising of 15*4=60 bytes.

a. The first element is the tilt compensated true compass heading coming out of the device.
b. Next 4 elements consisted of the delta vector i.e. the change in position x, y, z and the angle of rotation around z-axis (the change in the angle of the x-y plane). As these elements are of type float, thus are of 32 bits. The change in position is between two successive ZUPT instances, i.e. the displacement vector from one step to the next one.
c. The other 10 elements of the payload are used to form the covariance matrix, which is a 4x4 symmetric matrix, thus 10 elements. These are not used in the step-wise dead reckoning. These are used in data fusion from two MIMU22BL devices in case of dual foot mounted system.

3. **Step Counter**: Next two bytes consisted of step counter, which is a counter taking record of ZUPT instances observed. This is essentially number of times velocity of MIMU22BL goes down below predefined threshold. Hence it is the number of steps taken by the wearer, if MIMU22BL is used for foot-mounted pedestrian tracking.

4. **Checksum**: The last two bytes of 68 bytes are consisted of check sum which is sum of all the bytes received prior to these. These are used to cross-check correctness of the data transferred.

Step 2: ACK - Send acknowledgement for last data packet received from MIMU22BL

ACK consists of 5 bytes:

i. 1st byte: 01

ii. 2nd byte: P1 (First byte of data packet number of last data packet received)

iii. 3rd byte: P2 (Second byte of data packet number of last data packet received)

iv. 4th byte: Quotient of {(1+P1+P2) div 256}

v. 5th byte: Remainder of {(1+P1+P2) div 256}

Pseudo code for ACK generation:

```
i=0;
for(j=0; j<4; j++)
{
  header[j]=buffer[i++]& 0xFF; // 4-bytes assigned to HEADER; buffer is the 64 bytes/ 68 bytes data packet from MIMU22BL
}
packet_number_1=header[1]; // First byte of data packet number
packet_number_2=header[2]; // Second byte of data packet number
ack = createAck(ack,packet_number_1,packet_number_2); // Acknowledgement created
<code to send acknowledgement> //ACKNOWLEDGEMENT SENT


// HOW TO CREATE ACKNOWLEDGEMENT
// ACK consists of 5 bytes
createAck(byte[] ack, int packet_number_1, int packet_number_2)
{
        ack[0]=0x01; // 1st byte
        ack[1]= (byte)packet_number_1; // 2nd byte
        ack[2]= (byte)packet_number_2; // 3rd byte
        ack[3]= (byte)((1+packet_number_1+packet_number_2-(1+packet_number_1+packet_number_2) % 256)/256); // 4th byte – Quotient of {(1+P1+P2) div 256}
        ack[4]= (byte)((1+packet_number_1+packet_number_2) % 256); // 5th byte- Remainder of {(1+P1+P2) div 256}
        return ack;
}
```

Step 3: SWDR - Perform stepwise dead reckoning (SWDR) on the received data

MIMU22BL transmits tracking data with respect to its own frame which itself is not fixed with respect to the user's global reference frame. Therefore the data should undergo rotational transformation before being presented to the end user in a global reference frame. (Here global refers to the coordinate axis in which the system is initialized.)

The four values of Payload that are position and heading change vector, i.e. dx, dy, dz, da (da is the change in angle of rotation about z-axis). These values are the output from the device at every ZUPT instance (*ZUPT instance = Foot at complete standstill = Step detection*). As mentioned earlier, each set of delta values describe movement between two successive steps. The delta values prior and after each ZUPT instance, are independent to each other as the system is reset every ZUPT, i.e. the delta values in one data packet is independent of data value in data packet transmitted just before, just after and all other. The position and heading change vector are with reference to the coordinate frame of last ZUPT instance. If SWDR with compass heading command is chosen then before dx, dy, dz, da there will be four bytes of compass heading "dh" which should be printed as it is.

The 'da' corresponds to the deviation in the orientation of MIMU22BL. It is rotation of the x-y plane about the z-axis, and z-axis is always aligned with the gravity. The da is thus used for updating the orientation of the system, and obtaining the global reference frame. The two dimensions of displacement vector (dx, dy) are transformed to the global frame by applying rotation and thus (x,y) in the global frame is obtained. As we do not consider any change in alignment in z-axis, updating the z coordinate is performed by simply adding present dz to the z obtained for previous ZUPT.

Thus x,y,z coordinates in the global reference frame are obtained at every ZUPT.
Pseudo code for construction of tracked path:

```
x_sw[0]=x_sw[1]=x_sw[2]=x_sw[3]=0.0; // Initialize once
//  (x_sw[0], x_sw[1], x_sw[2]) = Final (X,Y,Z) in the user's reference frame
// x_sw[3] = Cumulative change in angle around z-axis in the user's ref frame
//  x_sw[0], x_sw[1], x_sw[2] and x_sw[3] are float (IEEE754 Single precision 32-bits)
packet_number_1=header[1]; // First byte of data packet number
packet_number_2=header[2]; // Second byte of data packet number
//DO FOLLOWING AFTER SENDING THE ACKNOWLEDGEMENT
packet_number = packet_number_1*256 + packet_number_2; //PACKAGE NUMBER ASSIGNED
if(packet_number_old != packet_number) // Perform Stepwise Dead Reckoning on receiving new data packet
{
        for(j=0;j<4;j++) // Only first 4 4-bytes data packets are required
        dx[j]=(double)payload[j];// Refer Fig 1& 2 for Payload; dx[]-  float (IEEE754 Single precision 32-bits)
```

```
                                        // (dx[0],dx[1],dx[2]) = Displacement in (x,y,z) in MIMU22BL's reference
frame
                                        // dx[3] = Change in angle around z axis, as indicated by MIMU22BL
 // Note: The first position of MIMU22BL defines origin of user's reference frame. This means that the start point
would always appear as origin in the user's reference frame
        stepwise_dr_tu();     // Perform Stepwise Dead Reckoning on the received data
        packet_number_old=packet_number;
}

// STEPWISE DEAD RECKONING
void stepwise_dr_tu()
{
        sin_phi=(float) Math.sin(x_sw[3]);  //
        cos_phi=(float) Math.cos(x_sw[3]); //
        delta[0]=cos_phi*dx[0]-sin_phi*dx[1]; // Transform / rotate two dimensional displacement vector (dx[0],
dx[1]) to the user's  reference frame
        delta[1]=sin_phi*dx[0]+cos_phi*dx[1]; // Transform / rotate two dimensional displacement vector (dx[0],
dx[1]) to the user's  reference frame
        delta[2]=dx[2]; // Assuming only linear changes along z-axis
        x_sw[0]+=delta[0]; // Final X in the user's reference frame
        x_sw[1]+=delta[1]; // Final Y in the user's reference frame
        x_sw[2]+=delta[2]; // Final Z in the user's reference frame
        x_sw[3]+=dx[3]; // Cumulative change in orientation in the user's reference frame
        distance+=Math.sqrt((delta[0]*delta[0]+delta[1]*delta[1]+delta[2]*delta[2])); // Distance of the tracked
data path
}
```

Step 4: Application platform acknowledges receiving last DATA packet by sending appropriate ACK to "MIMU22BL".

(Cycle of steps 3 and 4 is repeated until the application sends STOP. On receiving STOP command, MIMU22BL executes Step 5)

Follow the steps laid down in Step 2 for generating ACK for the received DATA packet.

Step 5: STOP - (i) Stop processing in MIMU22BL (ii) Stop all outputs in MIMU22BL

(i) Stop Processing in MIMU22BL by sending 3 bytes command: {0x32, 0x00, 0x32}

(ii) Stop all outputs in MIMU22BL by sending 3 bytes command : {0x22, 0x00, 0x22}