INDIAN INSTITUTE OF SPACE SCIENCE AND TECHNOLOGY

AV431: NAVIGATION LAB

Department of Avionics

Group Project: Foot Mounted Inertial Navigation System

Group Members:
Bijoy Mondal, SC19B130
Karun Mathews Manoj, SC19B120

Submitted to:
Dr. Priyadarshnam

Foot Mounted Inertial Navigation System

Group Project

December 13, 2022

Abstract

This project attempts to replicate the results by Nillson, JO et al [1],[2]. Nillson et al, developed a foot-mounted INS, and used the zero-velocity update algorithm as one method to reduce drift in estimated position and velocity values- which is due to the bias, instability and misalignment in the accelerometer and gyroscope. In our project, we estimate position and velocity by simple integration (only translation is considered). Zero-velocity update has also been implemented. The platform used was an android mobile, for which an app was developed using flutter.

1 Introduction

Development of an inertial navigation system, with high accuracy, robustness and no external infrastructure required is highly desirable for military or security personnel, and first responders. Such people may operate in locations without Satellite navigation support or in situations where delayed communication due to poor network coverage is not admissible. Therefore, at least for the short term, inertial navigation systems based on dead reckoning are reliable. Foot-mounted inertial navigation, aided by zero-velocity updates is an example of such a system.

Types of error:
- bias
- bias stability
- axis alignment

2 Methodology

2.1 Calibration

Procedure:

1. The aim of this process to ascertain the amount of bias present for the accelerometer sensor along x,y and z axes.

2. For this we measure the accelerometer reading along each axes for the +0g,1g,-0g and -1g positions. This corresponds to rotating the axis in steps of 90 degrees in the plane vertical to the ground.
3. 4 sets of 2 samples (whose average is used in the table) each are taken for each reading. The different sets are obtained by rotating the device to different orientations, while keeping that particular axis pointed in the same direction.
4. For each reading, an 8 sample average is obtained. This is used to calculate the calibration coefficients.
5. The following calibration coefficients are calculated: bias at 0g, bias at 1g, bias discrepancy and misalignment.
6. The bias for each axis is compensated by first checking whether magnitude of accelerometer reading lies closer to 0g or 1g- and then subtracting the corresponding bias from the accelerometer reading.

Below are the tables for x axis, y axis and z axis calibration.

Position	1	2	3	4	**8 sample avg (m/s^2)**
+0g (0°)	0.08025	0.465	-0.06	0.44	0.2313125
+1g (90°)	9.76	9.77	9.79	9.79	9.7775
-0g (180°)	0.09	-0.49	-0.08	-0.47	-0.2375
-1g (270°)	-9.79	-9.79	-9.78	-9.78	-9.785

Table 1: x axis calibration table

Position	1	2	3	4	**8 samples average (m/s^2)**
+0g (0°)	0.46	0.44	0.07	-0.105	-0.21625
+1g (90°)	9.76	9.76	9.8	9.8	9.78
-0g (180°)	-0.48	-0.46	0.05	-0.125	-0.25375
-1g (270°)	-9.78	-9.78	-9.77	-9.77	-9.7775

Table 2: y axis calibration table

Position	1	2	3	4	**8 samples average (m/s^2)**
+0g (0°)	-0.05	-0.045	0.01	-0.085	-0.02
+1g (90°)	9.77	9.785	9.785	9.78	9.78
-0g (180°)	-0.005	-0.095	-0.105	-0.01	-0.215
-1g (270°)	-9.85	-9.85	-9.85	-9.85	-9.85

Table 3: z axis calibration table

This is the code sample in which the bias is compensated for:

```
//x,y,z are the accelerometer readings
d1 = (x).abs();
d2 = (d1-9.8).abs();
if(d1<d2){
  x = x+ 0.00309375 ;
}else{
  x = x + 0.00375;
}

d1 = (y).abs();
d2 = (d1-9.8).abs();
if(d1<d2){
  y = y+ 0.235;
}else{
  y = y - 0.0025;
}

d1 = (z).abs();
d2 = (d1-9.8).abs();
if(d1<d2){
  z = z+ 0.1175;
}else{
  z = z + 0.01;
}
```

2.2 Foot-mounted setup and trial runs

In the above figure, the foot mounted setup is shown. The z axis is mostly aligned in the backwards direction, the x axis is mostly aligned towards the left and y axis points straight up along the foot (away from the ground).

The trial runs were conducted with the phone strapped as shown in the figure. Each trial run was for 20 s, and the sensor readings were taken every 100 ms- giving 200 data points. The trial runs were conducted by walking only straight ahead, on level ground and at a walking pace.

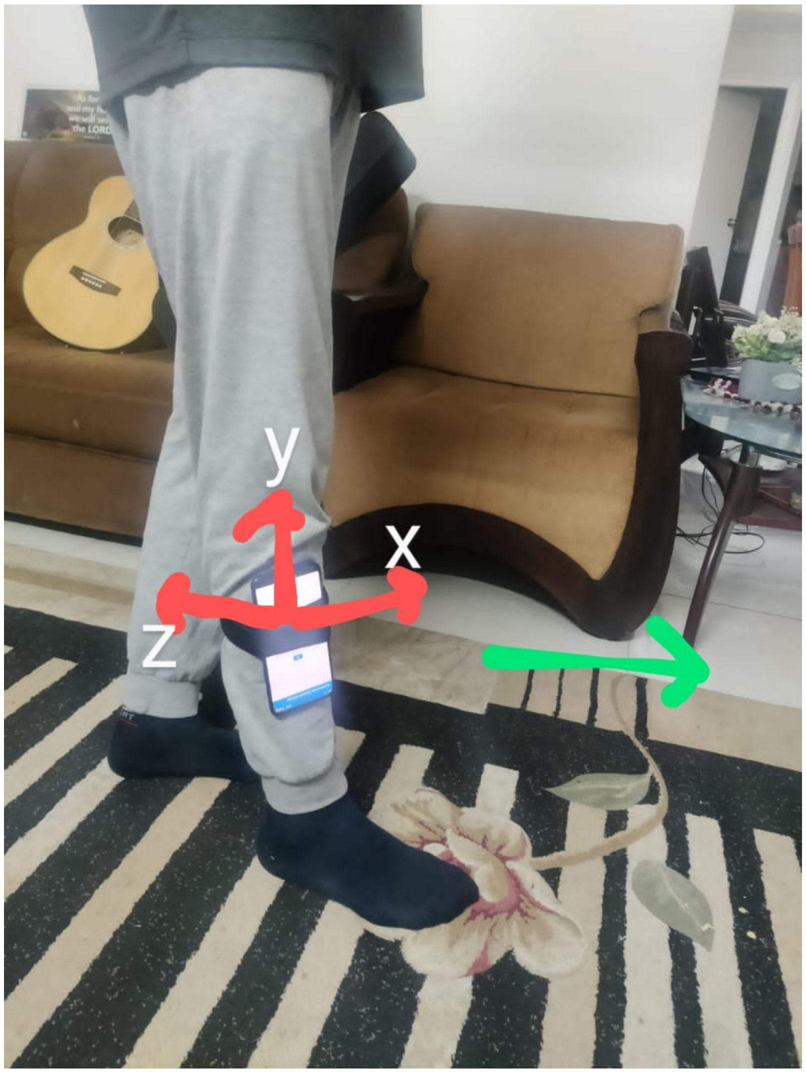
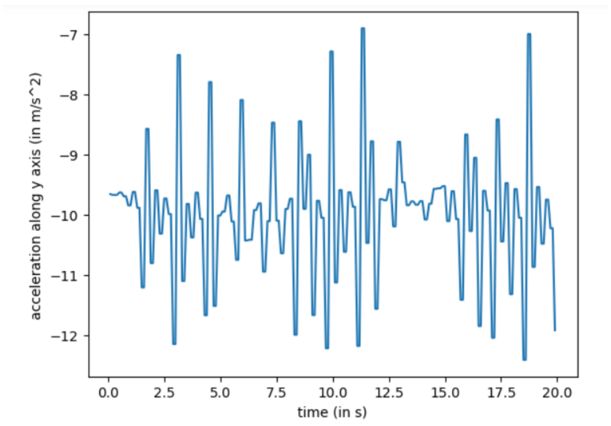
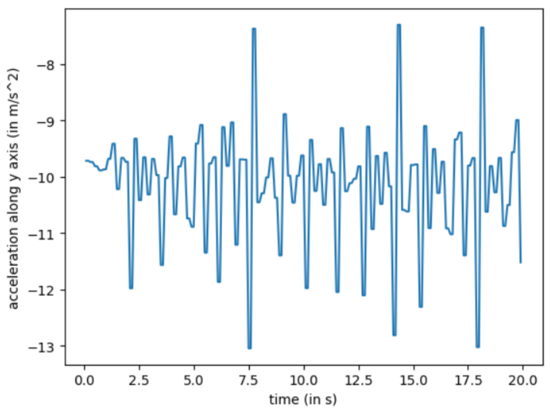


Figure 1: Foot mounted setup



(a) trial run 1



(b) trian run 2

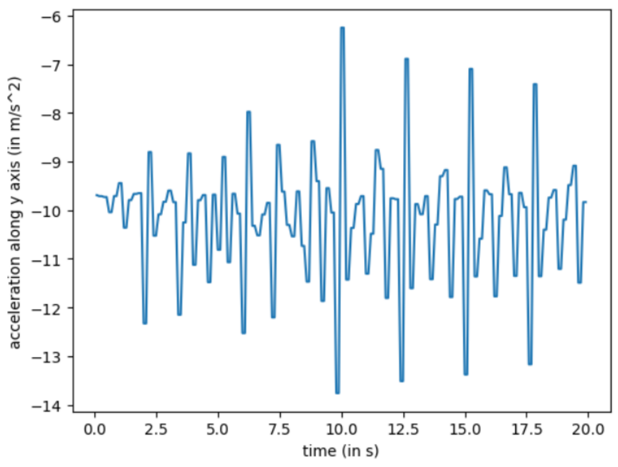


Figure 2: trial run 3

From each run the profile of y acceleration over time was observed, and a rough threshold was taken for detecting footfalls in the run. Using this the zero-velocity update algorithm was implemented.

2.3 Zero-velocity update (ZUPT)

We have used a very simple zero-velocity update method. In each run, it is observed if the acceleration along y axis crosses the threshold- then the x, y and z velocity variables are made zero. This will make the position constant for a short instant, and will stop some of the drift which was accumulating.

Above the y acceleration profile over time for three runs are shown. By looking at these, a threshold of -9.5 was assumed.

This is the code sample used for zero velocity update:

```
if(y>-9.5 && _selections[1]){
  xVel=0;
  yVel=0;
  zVel=0;
}else{
  xVel=xVel + x*delT;
  yVel=yVel + y*delT;
  zVel=zVel + z*delT;
}

xPos=xPos + xVel*delT;
yPos=yPos + yVel*delT;
zPos=zPos + zVel*delT;
```

3 Results and discussion

The calibration coefficients and standard deviation tables are given below:

Parameters	**Equations**	**x axis**	**y axis**	**z axis**
Bias at 0g ($\mu m/s^2$)	**((E0+E180)*10ˆ6)/2**	-3093.75	-235000	-117500
Bias at 0g ($\mu m/s^2$)	**((E90+E270)*10ˆ6)/2**	-3750	2500	-10000
Bias * Discrepancy ($\mu m/s^2$)	**Bias at 1g-Bias at 0g**	-656.25	237500	107500
Misalignment (arc Sec)	**((E0-E180)*10ˆ6)/(2*4.84)**	1430	3870	20140

Table 4: Calibration coefficients

	x axis	**y axis**	**z axis**
+0g (0°)	0.261943	0.279326	0.058452
+1g (90°)	0.015	0.023094	0.007071
-0g (180°)	0.288603	0.259852	0.0536
-1g (270°)	0.005774	0.005774	0

Table 5: Standard deviation table

Here we compare the trajectories obtained for the trial runs with bias compensation and ZUPT (zero-velocity update), with only bias compensation and with no bias compensation nor ZUPT (no error correction).

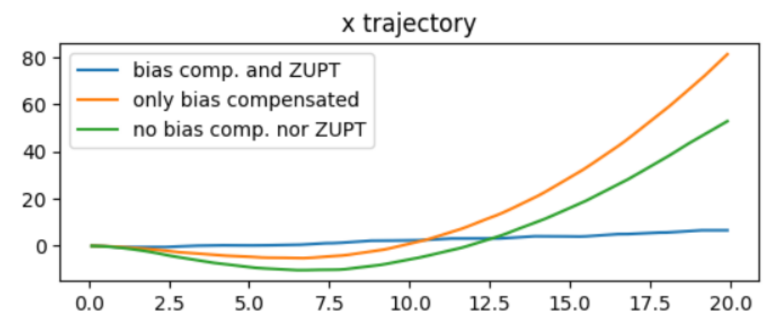


Figure 3: x trajectory comparison

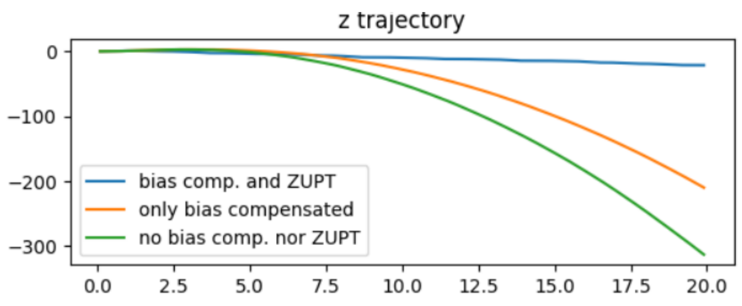


Figure 4: z trajectory comparison

Here the x and z trajectory are zoomed in and shown.

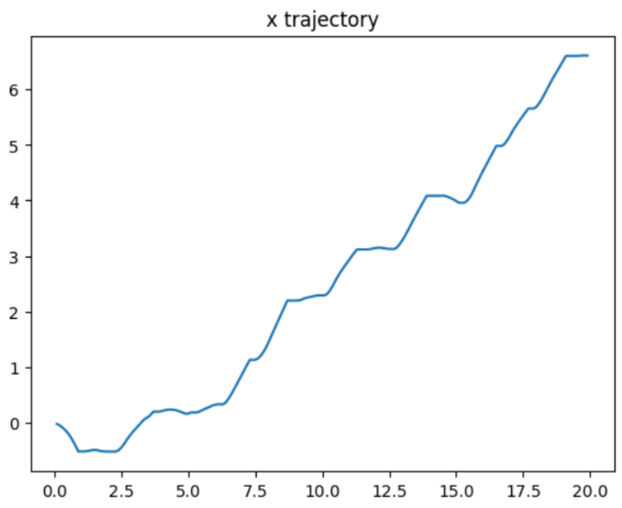


Figure 5: x trajectory

The trajectory profiles of the runs with ZUPT all have some common characteristics. It is seen that there are various points in the trajectory, where the slope becomes zero- and soon after that the slope starts increasing or decreasing. This gives the profile a bumpy pattern. These are the points where footfall is detected, and all the velocities are set to zero.

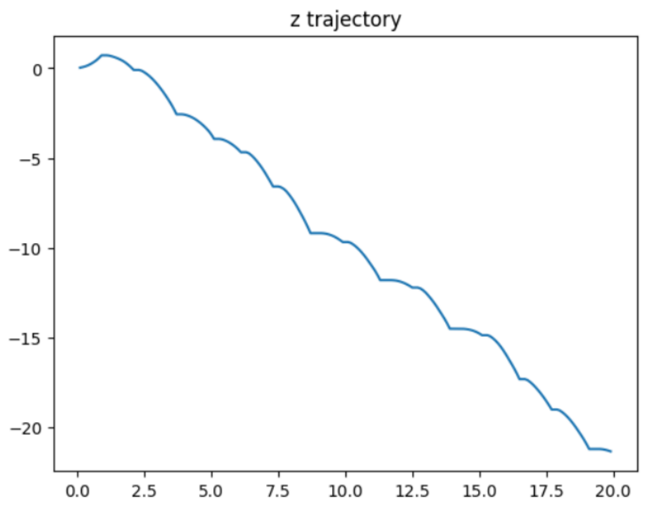


Figure 6: z trajectory

Also clearly from the comparison plots, we see the zero velocity update algorithm prevents the error from growing very large (by 10 or 100 times more than the expected value).

4 Conclusion

We have implemented a foot-mounted inertial navigation system with zero-velocity update algorithm. We have also shown that the zero-velocity update algorithm reduces a significant amount of error.

For future work, the algorithm should not just a threshold, since for different walking patterns, the threshold can change. Also, rotation of the body can also be considered- to get a more complete inertial navigation system.

5 Codes and User Manual

5.1 main.dart file

```
import 'dart:math';
import 'dart:async';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:sensors_plus/sensors_plus.dart';

import 'package:to_csv/to_csv.dart' as exportCSV;

void main() => runApp(MyApp());
class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'imu_appv2',
      theme: ThemeData(

        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(),
    );
  }
}
class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
}
class _MyHomePageState extends State<MyHomePage> {
  double x=0, y=0, z=0, t=0;
  double xGyro=0, yGyro=0, zGyro=0;
  double xVel=0, yVel=0, zVel=0;
  double xPos=0, yPos=0, zPos=0;
  double d1=0,d2=0;
  //bool genCSV=false;

  List<List<String>> trajectoryPts = [];
  List<String> data1=[];

  List<String> header = ['x','y','z','xAcc','yAcc','zAcc'];

  @override
```

```
void doIntegration(){
  t=0;
  xVel=0;yVel=0;zVel=0;
  xPos=0;yPos=0;zPos=0;
  trajectoryPts = [];

  const dt = Duration(milliseconds:100);
  double delT=0.1;
  double trackingPeriod=20;
  Timer.periodic(dt, (timer) {
    t=t+delT;

    //from the plots taken we see if yAcc > -9.5. it seems to
       indicate that the foot is on the ground
    //so every time yAcc > -9.5. all velocities are reset to zero
       .
    //this should eliminate some amount of drift


    if(y>-9.5 && _selections[1]){
      xVel=0;
      yVel=0;
      zVel=0;
    }else{
      xVel=xVel + x*delT;
      yVel=yVel + y*delT;
      zVel=zVel + z*delT;
    }

    xPos=xPos + xVel*delT;
    yPos=yPos + yVel*delT;
    zPos=zPos + zVel*delT;

    data1 = [xPos.toStringAsFixed(5),yPos.toStringAsFixed(5),zPos
       .toStringAsFixed(5),x.toStringAsFixed(5),y.toStringAsFixed
       (5)];
    trajectoryPts.add(data1);

    if(t>trackingPeriod){
      if(_selections[0]){
        exportCSV.myCSV(header,trajectoryPts);
      }
      timer.cancel();
    }
  });
}
```

```
List<bool> _selections = List.generate(2, (_)=>false);

void initState() {
  // TODO: implement initState
  super.initState();
  accelerometerEvents.listen((AccelerometerEvent event) {
    setState(() {
      x = event.x;
      y = event.y;
      z = event.z;

      d1 = (x).abs();
      d2 = (d1-9.8).abs();
      if(d1<d2){
        x = x+ 0.00309375 ;
      }else{
        x = x + 0.00375;
      }

      d1 = (y).abs();
      d2 = (d1-9.8).abs();
      if(d1<d2){
        y = y+ 0.235;
      }else{
        y = y - 0.0025;
      }

      d1 = (z).abs();
      d2 = (d1-9.8).abs();
      if(d1<d2){
        z = z+ 0.1175;
      }else{
        z = z + 0.01;
      }
    });
  });//get the sensor data and set then to the data types
  gyroscopeEvents.listen((GyroscopeEvent event) {
    setState(() {
      xGyro = event.x;
      yGyro = event.y;
      zGyro = event.z;
    });
  });//get the sensor data and set then to the data types
}
@override
```

```
Widget build(BuildContext context) {
  return Scaffold(
      appBar: AppBar(
        title: Text("Accelerometer and Gyro readings"),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            ToggleButtons(
                children: [
                  Icon(Icons.file_download),
                  Icon(Icons.run_circle_outlined)
                ],
                isSelected: _selections,
                onPressed: (int index) {
                  setState(() {
                    _selections[index] = !_selections[index];
                  });
                },
                color: Colors.grey,
                selectedColor: Colors.blue
            ),
            Text("Get csv! ... Strapped to feet?", style:
               TextStyle(fontSize: 13, fontStyle: FontStyle.
               normal),),
            ElevatedButton(
              child: Text("[0]", style: TextStyle(fontSize: 15),)
                 ,
              onPressed: doIntegration
            ),
            Text(t.toStringAsFixed(5), style: TextStyle(fontSize:
                13, fontStyle: FontStyle.normal),),
            Text("Current xyz coordinates:("+xPos.toStringAsFixed
               (5)+","+yPos.toStringAsFixed(5)+","+zPos.
               toStringAsFixed(5)+")", style: TextStyle(fontSize:
                13, fontStyle: FontStyle.normal),),
            Text("velocity along xyz axes:("+xVel.toStringAsFixed
               (5)+","+yVel.toStringAsFixed(5)+","+zVel.
               toStringAsFixed(5)+")", style: TextStyle(fontSize:
                13, fontStyle: FontStyle.normal),),
            Padding(
              padding: const EdgeInsets.all(10.0),
              child: Text(
                "Accelerometer readings:",
                style: TextStyle(fontSize: 18.0, fontWeight:
                   FontWeight.w900),
              ),
            ),
            Table(
              border: TableBorder.all(
                  width: 2.0,
                  color: Colors.blueAccent,
                  style: BorderStyle.solid),
              children: [
                TableRow(
                  children: [
                    Padding(
                      padding: const EdgeInsets.all(8.0),
                      child: Text(
                        "X axis : ",
                        style: TextStyle(fontSize: 20.0),
                      ),
                    ),
                    Padding(
                      padding: const EdgeInsets.all(8.0),
                      child: Text(x.toStringAsFixed(2), //trim
                         the asis value to 2 digit after decimal
                         point
                          style: TextStyle(fontSize: 20.0)),
                    )
                  ],
                ),
                TableRow(
                  children: [
                    Padding(
                      padding: const EdgeInsets.all(8.0),
                      child: Text(
                        "Y Axis : ",
                        style: TextStyle(fontSize: 20.0),
                      ),
                    ),
                    Padding(
                      padding: const EdgeInsets.all(8.0),
                      child: Text(y.toStringAsFixed(2),  //trim
                         the asis value to 2 digit after decimal
                         point
                          style: TextStyle(fontSize: 20.0)),
                    )
                  ],
                ),
                TableRow(
                  children: [
                    Padding(
                      padding: const EdgeInsets.all(8.0),
                      child: Text(
                        "Z Axis : ",
                        style: TextStyle(fontSize: 20.0),
                      ),
                    ),
                    Padding(
                      padding: const EdgeInsets.all(8.0),
                      child: Text(z.toStringAsFixed(2),   //trim
                         the asis value to 2 digit after decimal
                         point
                          style: TextStyle(fontSize: 20.0)),
                    )
                  ],
                ),
              ],
            ),
            Padding(
              padding: const EdgeInsets.all(10.0),
              child: Text(
                "Gyroscope readings:",
                style: TextStyle(fontSize: 18.0, fontWeight:
                   FontWeight.w900),
              ),
            ),
            Table(
              border: TableBorder.all(
                  width: 2.0,
                  color: Colors.blueAccent,
                  style: BorderStyle.solid),
              children: [
                TableRow(
                  children: [
                    Padding(
                      padding: const EdgeInsets.all(8.0),
                      child: Text(
                        "X axis : ",
                        style: TextStyle(fontSize: 20.0),
                      ),
                    ),
                    Padding(
                      padding: const EdgeInsets.all(8.0),
                      child: Text(xGyro.toStringAsFixed(2), //
                         trim the asis value to 2 digit after
                         decimal point
                          style: TextStyle(fontSize: 20.0)),
                    )
                  ],
                ),
                TableRow(
                  children: [
                    Padding(
                      padding: const EdgeInsets.all(8.0),
                      child: Text(
                        "Y Axis : ",
                        style: TextStyle(fontSize: 20.0),
                      ),
                    ),
                    Padding(
                      padding: const EdgeInsets.all(8.0),
                      child: Text(yGyro.toStringAsFixed(2),  //
                         trim the asis value to 2 digit after
                         decimal point
                          style: TextStyle(fontSize: 20.0)),
                    )
                  ],
                ),
                TableRow(
                  children: [
                    Padding(
                      padding: const EdgeInsets.all(8.0),
                      child: Text(
                        "Z Axis : ",
                        style: TextStyle(fontSize: 20.0),
                      ),
                    ),
                    Padding(
                      padding: const EdgeInsets.all(8.0),
                      child: Text(zGyro.toStringAsFixed(2),   //
                         trim the asis value to 2 digit after
                         decimal point
                          style: TextStyle(fontSize: 20.0)),
                    )
                  ],
                ),
              ],
            ),
          ],
              ),
            ));
      }
}
```

5.2 pubspec.yaml file

```
    name: imu_appv3
description: A new Flutter project.

# The following line prevents the package from being accidentally
   published to
# pub.dev using ‘flutter pub publish‘. This is preferred for
   private packages.
publish_to: 'none' # Remove this line if you wish to publish to pub
   .dev

# The following defines the version and build number for your
   application.
# A version number is three numbers separated by dots, like 1.2.43
# followed by an optional build number separated by a +.
# Both the version and the builder number may be overridden in
   flutter
# build by specifying --build-name and --build-number, respectively
   .
# In Android, build-name is used as versionName while build-number
   used as versionCode.
# Read more about Android versioning at https://developer.android.
   com/studio/publish/versioning
# In iOS, build-name is used as CFBundleShortVersionString while
   build-number is used as CFBundleVersion.
# Read more about iOS versioning at
# https://developer.apple.com/library/archive/documentation/General
   /Reference/InfoPlistKeyReference/Articles/CoreFoundationKeys.
   html
# In Windows, build-name is used as the major, minor, and patch
   parts
# of the product and file versions while build-number is used as
   the build suffix.
version: 1.0.0+1

environment:
  sdk: '>=2.18.5 <3.0.0'

# Dependencies specify other packages that your package needs in
```

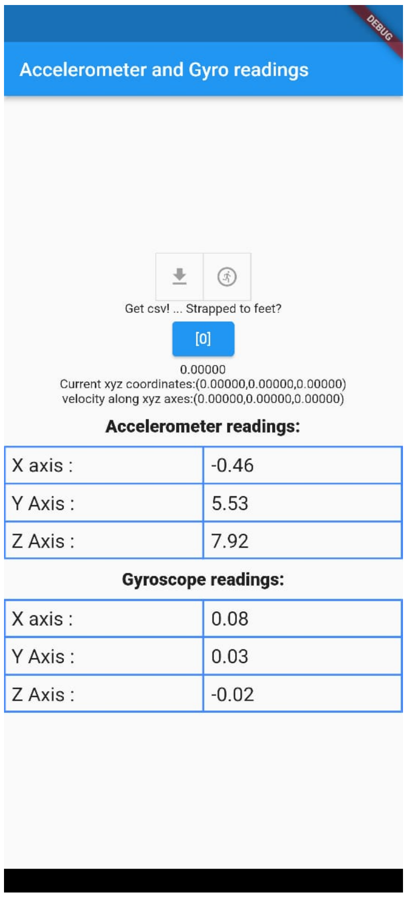
```
   order to work.
# To automatically upgrade your package dependencies to the latest
   versions
# consider running ‘flutter pub upgrade --major-versions‘.
   Alternatively,
# dependencies can be manually updated by changing the version
   numbers below to
# the latest version available on pub.dev. To see which
   dependencies have newer
# versions available, run ‘flutter pub outdated‘.
dependencies:
  flutter:
    sdk: flutter

  sensors_plus: "^2.0.1"

  to_csv: "^1.1.1"

  # The following adds the Cupertino Icons font to your application
     .
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.2

dev_dependencies:
  flutter_test:
    sdk: flutter

  # The "flutter_lints" package below contains a set of recommended
      lints to
  # encourage good coding practices. The lint set provided by the
     package is
  # activated in the ‘analysis_options.yaml‘ file located at the
     root of your
  # package. See that file for information about deactivating
     specific lint
  # rules and activating additional ones.
  flutter_lints: ^2.0.0

# For information on the generic Dart part of this file, see the
# following page: https://dart.dev/tools/pub/pubspec

# The following section is specific to Flutter packages.
flutter:

  # The following line ensures that the Material Icons font is
  # included with your application, so that you can use the icons
```

```
      in
  # the material Icons class.
  uses-material-design: true
```

5.3 functions and classes used

1. Toggle Buttons class.
2. Timer class, Timer.periodic() function, Duration class
3. to_csv package
4. sensors_plus package

5.4 the app


DEBUG
Accelerometer and Gyro readings
Get csv! ... Strapped to feet?
[0]

0.00000
Current xyz coordinates:(0.00000,0.00000,0.00000)
velocity along xyz axes:(0.00000,0.00000,0.00000)

Accelerometer readings:

X axis :	-0.46
Y Axis :	5.53
Z Axis :	7.92

Gyroscope readings:

X axis :	0.08
Y Axis :	0.03
Z Axis :	-0.02

Figure 7: screenshot of the app

The download button is for getting csv file of the trajectory, and the walking symbol toggle button is to be toggled on when strapped to the feet (then ZUPT is used).

References

[1] L. Xing, X. Tu, and Z. Chen, "Foot-mounted pedestrian navigation method by comparing adr and modified zupt based on mems imu array," *Sensors*, vol. 20, no. 13, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/13/3787

[2] J. Nilsson, D. Zachariah, and I. e. a. Skog, "Cooperative localization by dual foot-mounted inertial sensors and inter-agent ranging," *EURASIP J. Adv. Signal Process*, 2013. [Online]. Available: https://asp-eurasipjournals.springeropen.com/articles/10.1186/1687-6180-2013-164#citeas

[3] R. Palankar, "Flutter sensors plugin – accelerometer and gyroscope sensor library." [Online]. Available: https://protocoderspoint.com/flutter-sensors-plugin-accelerometer-and-gyroscope-sensors-library/