EMF App

- Playlist I referred :

  https://www.youtube.com/playlist?list=PL9n0l8rSshSl8iwG9fUpvUZcNWxeeFhxJ

- EMF - **ElectroMotive Force** which can be calculated using the magnetometer sensor which is already present in our phones.
- We can view emf values for all the three axes, x, y and z axis.
- For state management, Provider state management is used.

▼ Folders

`main.dart`

- Here, a Provider named `MagnitudeProvider()` is created, which will notify about the changes in the app.
- we also return `Home()` page in the main.dart file as it will return the home screen.

`home.dart`

- first, a simple scaffold would be defined, as it would be the base of the screen
- We use `SingleChildScrollView()` widget to make it scrollable for smaller screens.
- We put all the elements inside a column.
- The elements are →
  - `MainReading()`

    for showing the net field in μ Tesla

    - `XYZReading()`

      to show live x, y and z fields' magnetic field

    - `MeterReading()`

      To show the gauge

    - `ElevatedButton()`

      It is the '**visualise**' button, which, `onPressed: {}` shows another screen, Visuals, which shows the graphs of the magnetic fields

    - `Consumer<MagnitudeProvider>`

      MagnitudeProvider has a function, `changeValues();` which after clicking the '**start**' button (which is a child of ConsumerProvider widget), will start capturing the values provided by the magnitudeprovider.

- Widgets

  `mainReading.dart`

    - This function defines how the first widget is going to look like, on a home page.
    - as the values are going to change very fast, the µTesla written after will be shaky, that's why we've used two seperate containers.
    - we are updating the value with `magnitude` from `model`, i.e. with the data from magnetometer sensor.

  `xyzReading.dart`

    - we use three radio buttons, to set the speed at which we want the EMF data getting changed.
    - using the function `onchanged()`, the setUpdateInterval is called, and the speed of updating data is changed.
    - again, we use a sizedbox to display changing values of x, y, z by creating a consumer, which consumes values from `<magnitudeprovider>` and updates them accordingly.
    - inside that sizedbox, the three textwidgets update value of x, y and z.

  `textwidget.dart`

    - textwidget defines the font, size, etc. of texts used to display values of x, y and z(previous line)

  `meterReading.dart`

    - it contains a gauge widget, and its properties, like radial axis, maximum value, minimum value, offets, etc.
    - `GaugeRange` specifies the range (good, moderate, etc.)
    - `needlepointer` points to `magnitude`.
    - `GaugeAnnotation` shows `magnitude` as well.
- Models

  `magnitudeProvider.dart`

    - the class MagnitudeProvider is extending ChangeNotifier
    - x, y, z, magnitude are initialised as doubles
    - We also define an empty list of objects of class LiveData
    - vector3 is used for the data from sensors
    - groupvalue is how much FPS we want.
    - `changeValues()` function is accesed by the start button of home screen.
    - as soon as `changevalues()` function is triggered, it starts capturing values from motionsensors (i.e. `motionSensors.magnetometer.listen(MagnetometerEvent event)` .
    - the values get set when magnetometer values starts coming in.
    - magnitude value is calculated using :

      $$\sqrt{(x^2 + y^2 + z^2)}$$

    - using `values.add(LiveData(x,y,z, time++));` we are adding the values to list values, in order to plot a graph.
    - In order to maintain size of `values` list, we are removing the first element of values using `values.removeAt(0)`, after 40 elements get added to list.
- Utils

  `Colors.dart`

    - defining class AppColors → primary color, red, etc.
    - all colors are declared as static, because we need to access them from anywhere
- Pages

`visuals.dart`

    - `ChartSeriesController` is used if you want to make some changes in the graph.
    - Although we are not going to use it, we'll just define it.
    - We are using a card inside a container to show graph.
    - We again use a consumer of MagnitudeProvider, and We'll plot a graph using `SfCartesianChart`
    - The format of the graph is `LineSeries`.
    - It uses data from class `LiveData` and plots it on graph.
    - it plots graphs for values of x versus time, y versus time, z vs time.
    - this is defined using xvaluemapper, yvaluemapper.
    - primaryxaxis is used for defining details, intervals, etc. of the graph.

- Dependencies :
    - motion_sensors (to get data from sensors)
    - provider (change notifier etc.)
    - syncfusion_flutter_gauges
    - syncfusion_flutter_gauges