Main.dart

import 'pages/monitor.dart';

import 'package:flutter/material.dart';

import 'pages/alarm.dart';

import 'pages/motor.dart';

import 'package:flutter\_local\_notifications/flutter\_local\_notifications.dart';

import 'local\_notications\_helper.dart';

import 'package:firebase\_database/firebase\_database.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      title: 'Skripsi',

      theme: ThemeData(

        primarySwatch: Colors.pink,

      ),

      home: new RootPage(),

    );

  }

}

class RootPage extends StatefulWidget {

  @override

  \_RootPageState createState() => \_RootPageState();

}

class \_RootPageState extends State<RootPage>

    with SingleTickerProviderStateMixin {

  TabController \_tabController;

  final \_notifications = FlutterLocalNotificationsPlugin(); //notifikasi

  final \_fireBase = FirebaseDatabase.instance.reference();

  @override

  void initState() {

    super.initState();

    \_tabController = TabController(vsync: this, length: 3);

    final settingsAndroid = AndroidInitializationSettings('app\_icon');

    final settingsIOS = IOSInitializationSettings(

        onDidReceiveLocalNotification: (id, title, body, payload) =>

            onSelectNotification(payload));

    \_notifications.initialize(

        InitializationSettings(settingsAndroid, settingsIOS),

        onSelectNotification: onSelectNotification);

    notifikasi();

  }

  void notifikasi() {

    String titelAlarm;

    \_fireBase.child("alarm").onChildChanged.listen((event) {

      int \_numId;

      String \_value;

      if ( event.snapshot.key == "overheat") {

        \_numId = 0;

        titelAlarm = "Motor Alarm";

        \_value = 'Over heat motor';

      }

      if ( event.snapshot.key == "overload") {

        \_numId = 1;

        titelAlarm = "Motor Alarm";

        \_value = 'Over current motor';

      }

      if ( event.snapshot.key == "level") {

        \_numId = 2;

        titelAlarm = "Level Alarm";

        \_value = 'Low watter level';

      }

      if ( event.snapshot.key == "flow") {

        \_numId = 3;

        titelAlarm = "Flow Alarm";

        \_value = 'Low pressure';

      }

      if (event.snapshot.value == true) {

        showOngoingNotification(FlutterLocalNotificationsPlugin(),

            title: titelAlarm, body: "$\_value ", id: \_numId);

      }

    });

  }

  @override

  void dispose() {

    \_tabController.dispose();

    super.dispose();

  }

  Future onSelectNotification(String payload) async {

    \_tabController.animateTo(2);

    \_notifications.cancelAll();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        backgroundColor: Colors.blue,

        title: Text("Kendali Pompa Air"),

        bottom: TabBar(

          controller: \_tabController,

          tabs: <Widget>[

            new Tab(

              icon: new Icon(

                Icons.pan\_tool,

                color: Colors.purple,

              ),

              text: "Setting",

            ),

            new Tab(

              icon: new Icon(

                Icons.network\_check,

                color: Colors.orange[700],

              ),

              text: "Monitor",

            ),

            new Tab(

              icon: new Icon(

                Icons.notifications\_active,

                color: Colors.red,

              ),

              text: "Alarm",

            ),

          ],

        ),

      ),

      body: new TabBarView(

        controller: \_tabController,

        children: <Widget>[

          new Motor(),

          new Monitor(),

          new Alarm(),

        ],

      ),

    );

    // );

  }

}

local\_notications\_helper.dart

import 'package:flutter\_local\_notifications/flutter\_local\_notifications.dart';

import 'package:meta/meta.dart';

NotificationDetails get \_noSound {

  final androidChannelSpecifics = AndroidNotificationDetails(

    'silent channel id',

    'silent channel name',

    'silent channel description',

    playSound: false,

  );

  final iOSChannelSpecifics = IOSNotificationDetails(presentSound: false);

  return NotificationDetails(androidChannelSpecifics, iOSChannelSpecifics);

}

Future showSilentNotification(

  FlutterLocalNotificationsPlugin notifications, {

  @required String title,

  @required String body,

  int id = 0,

}) =>

    \_showNotification(notifications,

        title: title, body: body, id: id, type: \_noSound);

NotificationDetails get \_ongoing {

  final androidChannelSpecifics = AndroidNotificationDetails(

    'your channel id',

    'your channel name',

    'your channel description',

    importance: Importance.Max,

    priority: Priority.High,

    ongoing: true,

    autoCancel: false,

  );

  final iOSChannelSpecifics = IOSNotificationDetails();

  return NotificationDetails(androidChannelSpecifics, iOSChannelSpecifics);

}

Future showOngoingNotification(

  FlutterLocalNotificationsPlugin notifications, {

  @required String title,

  @required String body,

  int id = 0,

}) =>

    \_showNotification(notifications,

        title: title, body: body, id: id, type: \_ongoing);

Future \_showNotification(

  FlutterLocalNotificationsPlugin notifications, {

  @required String title,

  @required String body,

  @required NotificationDetails type,

  int id = 0,

}) =>

    notifications.show(id, title, body, type);

alarm.dart

import 'package:flutter/material.dart';

import 'package:firebase\_database/firebase\_database.dart';

class Alarm extends StatefulWidget {

  @override

  \_AlarmState createState() => \_AlarmState();

}

class \_AlarmState extends State<Alarm> {

  final dbAlarm = FirebaseDatabase.instance.reference();

  List<bool> status = [false, false, false, false];

  final List<String> normal = [

    'Normal current motor',

    'Normal temperature',

    'Normal water level',

    'Normal pressure'

  ];

  final List<String> alarm = [

    'Over current motor',

    'Over heat motor',

    'Low watter level',

    'Low pressure'

  ];

  final List<String> normalMsg = [

    'Arus motor dalam keadaan normal',

    'Suhu motor dalam keadaan normal',

    'Level air sumur normal',

    'Tekanan air normal'

  ];

  final List<String> alarmMsg = [

    'Arus motor berlebih, cehek motor lalu tekan reset',

    'Suhu motor berlebih, cehek motor lalu tekan reset',

    'Level air sumur normal, cehek level lalu tekan reset',

    'Tekanan air tidak normal, cehek motor dan level lalu tekan reset'

  ];

  int nilai;

  @override

  void initState() {

    super.initState();

    dbInit();

  }

  void dbInit() {

    //baca state mode dari firebase

    dbAlarm.child("alarm").once().then((DataSnapshot snapshot) {

      Map<dynamic, dynamic> values = snapshot.value;

      values.forEach((key, values) {

        if (key == "flow" && values != null) {

          setState(() {

            status[3] = values.toString() == "true" ? true : false;

          });

        }

        if (key == "level" && values != null) {

          setState(() {

            status[2] = values.toString() == "true" ? true : false;

          });

        }

        if (key == "overheat" && values != null) {

          setState(() {

            status[1] = values.toString() == "true" ? true : false;

          });

        }

        if (key == "overload" && values != null) {

          setState(() {

            status[0] = values.toString() == "true" ? true : false;

          });

        }

      });

    });

    //listener untuk mode

    // child("alarm").onChildChanged.listen((event)

    dbAlarm.child("alarm").onChildChanged.listen(

      (event) {

        print(event.snapshot.key.toString() +

            " " +

            event.snapshot.value.toString());

        if (event.snapshot.key == "flow" && event.snapshot.value != null) {

          setState(() {

            status[3] =

                event.snapshot.value.toString() == "true" ? true : false;

          });

        }

        if (event.snapshot.key == "level" && event.snapshot.value != null) {

          setState(() {

            status[2] =

                event.snapshot.value.toString() == "true" ? true : false;

          });

        }

        if (event.snapshot.key == "overheat" && event.snapshot.value != null) {

          setState(() {

            status[1] =

                event.snapshot.value.toString() == "true" ? true : false;

          });

        }

        if (event.snapshot.key == "overload" && event.snapshot.value != null) {

          setState(() {

            status[0] =

                event.snapshot.value.toString() == "true" ? true : false;

          });

        }

      },

    );

  }

  List<Widget> createItems() {

    List<Widget> items = List<Widget>();

    for (var i = 0; i < 4; i++) {

      if (status[i] == true) {

        items.add(

          Card(

            color: Colors.yellow[100],

            child: Column(

              mainAxisSize: MainAxisSize.min,

              children: <Widget>[

                ListTile(

                  leading: Icon(

                    Icons.cancel,

                    size: 50.0,

                    color: Colors.red,

                  ),

                  title: Text(this.alarm[i]),

                  subtitle: Text(this.alarmMsg[i]),

                ),

                ButtonBar(

                  children: <Widget>[

                    FlatButton(

                      child: const Text('Reset'),

                      onPressed: () {

                        if (i == 3) {

                          dbAlarm

                              .child("nodeGet")

                              .update({"flow\_reset": false});

                        }

                        print(i);

                        // widget.\_preseter.dbSetInt('V$i', 0);

                      },

                    ),

                  ],

                ),

              ],

            ),

          ),

        );

      } else {

        items.add(

          Card(

            color: Colors.blue[100],

            child: Column(

              mainAxisSize: MainAxisSize.min,

              children: <Widget>[

                ListTile(

                  leading: Icon(

                    Icons.check\_box,

                    size: 50.0,

                    color: Colors.green,

                  ),

                  title: Text(this.normal[i]),

                  subtitle: Text(this.normalMsg[i]),

                ),

              ],

            ),

          ),

        );

      }

      items.add(Container(height: 10.0));

    }

    return items;

  }

  @override

  Widget build(BuildContext context) {

    return ListView(

      padding: EdgeInsets.all(15.0),

      children: createItems(),

    );

  }

}

Monitor.dart

// import 'dart:convert';

import 'package:flutter/material.dart';

import 'package:json\_table/json\_table.dart';

import 'package:syncfusion\_flutter\_gauges/gauges.dart';

import 'package:firebase\_database/firebase\_database.dart';

import 'package:flutter\_speed\_dial/flutter\_speed\_dial.dart';

class Monitor extends StatefulWidget {

  @override

  \_MonitorState createState() => \_MonitorState();

}

class \_MonitorState extends State<Monitor> {

  final nodeMCU = FirebaseDatabase.instance.reference();

  List jsonSample;

  double sumurValue = 0;

  double tankvalue = 0;

  @override

  void initState() {

    super.initState();

    modelListen();

    dbInit();

  }

  void dbInit() {

    nodeMCU.child("nodeSet").once().then(

      (DataSnapshot snapshot) {

        Map<dynamic, dynamic> values = snapshot.value;

        values.forEach((key, val) {

          if (val != null) {

            if (key == "sumur") {

              setState(() {

                sumurValue = val.toDouble() \* 10;

              });

            }

            if (key == "tangki") {

              setState(() {

                tankvalue = val.toDouble() \* 10;

              });

            }

          }

        });

      },

    );

    nodeMCU.child("nodeSet").onChildChanged.listen(

      (event) {

        if (event.snapshot.value != null) {

          if (event.snapshot.key == "tangki") {

            setState(() {

              tankvalue = event.snapshot.value.toDouble() \* 10;

            });

          }

          if (event.snapshot.key == "sumur") {

            setState(() {

              sumurValue = event.snapshot.value.toDouble() \* 10;

            });

          }

        }

      },

    );

  }

  void modelListen() {

    nodeMCU.child("Log").once().then(

      (DataSnapshot snapshot) {

        Map<dynamic, dynamic> map = snapshot.value;

        List<dynamic> list = map.values.toList()

          ..sort((a, b) => b['Jam'].compareTo(a['Jam']));

        setState(() {

          jsonSample = list; //jsonDecode(newString) as List;

        });

      },

    );

  }

  void deleteLog() {

    nodeMCU.child("Log").remove();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: SingleChildScrollView(

        padding: EdgeInsets.all(16.0),

        child: Container(

          child: Column(

            children: [

              Row(

                mainAxisAlignment: MainAxisAlignment.spaceAround,

                children: <Widget>[

                  Text(

                    "Sumur",

                    textAlign: TextAlign.center,

                    style: TextStyle(fontSize: 30.0, color: Colors.blueAccent),

                  ),

                  Text(

                    "Tank",

                    textAlign: TextAlign.center,

                    style: TextStyle(fontSize: 30.0, color: Colors.blueAccent),

                  )

                ],

              ),

              Row(

                children: <Widget>[

                  Expanded(

                    child: Container(

                      color: Colors.black54, //beground

                      child: SfRadialGauge(

                        key: null,

                        axes: <RadialAxis>[

                          RadialAxis(

                              radiusFactor: 0.98,

                              startAngle: 140,

                              endAngle: 40,

                              minimum: 0,

                              maximum: 130,

                              showAxisLine: false,

                              majorTickStyle: MajorTickStyle(

                                  length: 0.15,

                                  lengthUnit: GaugeSizeUnit.factor,

                                  thickness: 2),

                              labelOffset: 8,

                              axisLabelStyle: GaugeTextStyle(

                                  fontFamily: 'Times',

                                  fontSize: 12,

                                  fontWeight: FontWeight.w800,

                                  fontStyle: FontStyle.italic),

                              minorTicksPerInterval: 9,

                              interval: 10,

                              pointers: <GaugePointer>[

                                NeedlePointer(

                                    value: sumurValue,

                                    needleStartWidth: 2,

                                    needleEndWidth: 2,

                                    needleColor: const Color(0xFFF67280),

                                    needleLength: 0.8,

                                    lengthUnit: GaugeSizeUnit.factor,

                                    enableAnimation: true,

                                    animationType: AnimationType.bounceOut,

                                    animationDuration: 1500,

                                    knobStyle: KnobStyle(

                                        knobRadius: 8,

                                        sizeUnit: GaugeSizeUnit.logicalPixel,

                                        color: const Color(0xFFF67280)))

                              ],

                              minorTickStyle: MinorTickStyle(

                                  length: 0.08,

                                  thickness: 1,

                                  lengthUnit: GaugeSizeUnit.factor,

                                  color: const Color(0xFFC4C4C4)),

                              axisLineStyle: AxisLineStyle(

                                  color: const Color(0xFFDADADA),

                                  thicknessUnit: GaugeSizeUnit.factor,

                                  thickness: 0.1)),

                        ],

                      ),

                    ),

                  ),

                  Expanded(

                    child: Container(

                      color: Colors.black54, //beground

                      child: SfRadialGauge(

                        key: null,

                        axes: <RadialAxis>[

                          RadialAxis(

                              radiusFactor: 0.98,

                              startAngle: 140,

                              endAngle: 40,

                              minimum: 0,

                              maximum: 130,

                              showAxisLine: false,

                              majorTickStyle: MajorTickStyle(

                                  length: 0.15,

                                  lengthUnit: GaugeSizeUnit.factor,

                                  thickness: 2),

                              labelOffset: 8,

                              axisLabelStyle: GaugeTextStyle(

                                  fontFamily: 'Times',

                                  fontSize: 12,

                                  fontWeight: FontWeight.w800,

                                  fontStyle: FontStyle.italic),

                              minorTicksPerInterval: 9,

                              interval: 10,

                              pointers: <GaugePointer>[

                                NeedlePointer(

                                    value: tankvalue,

                                    needleStartWidth: 2,

                                    needleEndWidth: 2,

                                    needleColor: const Color(0xFFF67280),

                                    needleLength: 0.8,

                                    lengthUnit: GaugeSizeUnit.factor,

                                    enableAnimation: true,

                                    animationType: AnimationType.bounceOut,

                                    animationDuration: 1500,

                                    knobStyle: KnobStyle(

                                        knobRadius: 8,

                                        sizeUnit: GaugeSizeUnit.logicalPixel,

                                        color: const Color(0xFFF67280)))

                              ],

                              minorTickStyle: MinorTickStyle(

                                  length: 0.08,

                                  thickness: 1,

                                  lengthUnit: GaugeSizeUnit.factor,

                                  color: const Color(0xFFC4C4C4)),

                              axisLineStyle: AxisLineStyle(

                                  color: const Color(0xFFDADADA),

                                  thicknessUnit: GaugeSizeUnit.factor,

                                  thickness: 0.1)),

                        ],

                      ),

                    ),

                  ),

                ],

              ),

              jsonSample == null

                  ? Center(

                      child: CircularProgressIndicator(),

                    )

                  : JsonTable(

                      jsonSample,

                      showColumnToggle: true,

                      allowRowHighlight: true,

                      rowHighlightColor: Colors.green.withOpacity(0.7),

                      paginationRowCount: 40,

                      onRowSelect: (index, map) {

                        print(index);

                        print(map);

                      },

                    ),

              SizedBox(

                height: 40.0,

              ),

              Text("Skripsi Kendali Pompa Air Berbasis IoT")

            ],

          ),

        ),

      ),

      floatingActionButton: \_getFAB(),

    );

  }

  Widget \_getFAB() {

    return SpeedDial(

      animatedIcon: AnimatedIcons.menu\_close,

      animatedIconTheme: IconThemeData(size: 22),

      backgroundColor: Color(0xFF801E48),

      visible: true,

      curve: Curves.bounceIn,

      children: [

        // FAB 1

        SpeedDialChild(

            child: Icon(Icons.delete\_forever),

            backgroundColor: Color(0xFF801E48),

            onTap: () {

              /\* do anything \*/

              deleteLog();

            },

            label: 'Reset Log',

            labelStyle: TextStyle(

                fontWeight: FontWeight.w500,

                color: Colors.white,

                fontSize: 16.0),

            labelBackgroundColor: Color(0xFF801E48)),

        // FAB 2

        SpeedDialChild(

            child: Icon(Icons.refresh),

            backgroundColor: Color(0xFF801E48),

            onTap: () {

              setState(() {

                jsonSample = null;

                modelListen();

                // \_counter = 0;

              });

            },

            label: 'Refres Log',

            labelStyle: TextStyle(

                fontWeight: FontWeight.w500,

                color: Colors.white,

                fontSize: 16.0),

            labelBackgroundColor: Color(0xFF801E48))

      ],

    );

  }

}

Motor.dart

import 'package:flutter/material.dart';

import 'package:firebase\_database/firebase\_database.dart';

class Motor extends StatefulWidget {

  @override

  \_MotorState createState() => \_MotorState();

}

class \_MotorState extends State<Motor> {

  double suhu = 0;

  double flow = 0;

  var warna = Colors.green[300];

  final sumurMax = TextEditingController();

  final sumurMin = TextEditingController();

  final tangkiMax = TextEditingController();

  final tangkiMin = TextEditingController();

  final \_formKey = GlobalKey<FormState>();

  final dbRef = FirebaseDatabase.instance.reference();

  @override

  void initState() {

    super.initState();

    dbInit();

    sumurMax.addListener(\_setSumurMax);

    sumurMin.addListener(\_setSumurMin);

    tangkiMax.addListener(\_setTangkiMax);

    tangkiMin.addListener(\_setTangkiMin);

  }

  \_setSumurMax() {

    print(sumurMax.text);

  }

  \_setSumurMin() {

    print(sumurMin.text);

  }

  \_setTangkiMax() {

    print(tangkiMax.text);

  }

  \_setTangkiMin() {

    print(tangkiMin.text);

  }

  @override

  void dispose() {

    sumurMax.dispose();

    sumurMin.dispose();

    tangkiMax.dispose();

    tangkiMin.dispose();

    super.dispose();

  }

  void dbInit() {

    //baca state mode dari firebase

    dbRef.child("nodeSet").once().then((DataSnapshot snapshot) {

      Map<dynamic, dynamic> values = snapshot.value;

      values.forEach((key, values) {

        if (key == "flow" && values != null) {

          setState(() {

            flow = values.toDouble();

          });

        }

        if (key == "suhu" && values != null) {

          setState(() {

            suhu = values.toDouble();

          });

        }

      });

    });

    //listener untuk mode

    dbRef.child("nodeSet").onChildChanged.listen(

      (event) {

        if (event.snapshot.value != null) {

          if (event.snapshot.key == "flow") {

            setState(() {

              flow = event.snapshot.value.toDouble();

            });

          }

          if (event.snapshot.key == "suhu") {

            setState(() {

              suhu = event.snapshot.value.toDouble();

            });

            if (suhu < 60) {

              setState(() {

                warna = Colors.green[300];

              });

            } else if (suhu >= 100) {

              setState(() {

                warna = Colors.red[300];

              });

            } else {

              setState(() {

                warna = Colors.amber;

              });

            }

          }

        }

      },

    );

    dbRef.child("nodeGet").once().then((DataSnapshot snapshot) {

      Map<dynamic, dynamic> values = snapshot.value;

      values.forEach((key, values) {

        if (key == "sumur\_off\_level" && values != null) {

          setState(() {

            sumurMin.text = values.toString();

          });

        }

        if (key == "sumur\_on\_level" && values != null) {

          setState(() {

            sumurMax.text = values.toString();

          });

        }

      });

    });

    //listener untuk mode

    dbRef.child("nodeGet").onChildChanged.listen(

      (event) {

        if (event.snapshot.key == "sumur\_off\_level" &&

            event.snapshot.value != null) {

          setState(() {

            sumurMin.text = event.snapshot.value.toString();

          });

        }

        if (event.snapshot.key == "sumur\_on\_level" &&

            event.snapshot.value != null) {

          setState(() {

            sumurMax.text = event.snapshot.value.toString();

          });

        }

      },

    );

    dbRef.child("tankGet").once().then((DataSnapshot snapshot) {

      Map<dynamic, dynamic> values = snapshot.value;

      values.forEach((key, values) {

        if (key == "tangki\_min" && values != null) {

          setState(() {

            tangkiMin.text = values.toString();

          });

        }

        if (key == "tangki\_max" && values != null) {

          setState(() {

            tangkiMax.text = values.toString();

          });

        }

      });

    });

    //listener untuk mode

    dbRef.child("tankGet").onChildChanged.listen(

      (event) {

        if (event.snapshot.key == "tangki\_min" &&

            event.snapshot.value != null) {

          setState(() {

            tangkiMin.text = event.snapshot.value.toString();

          });

        }

        if (event.snapshot.key == "tangki\_max" &&

            event.snapshot.value != null) {

          setState(() {

            tangkiMax.text = event.snapshot.value.toString();

          });

        }

      },

    );

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: Center(

        child: SingleChildScrollView(

          child: Column(

            children: <Widget>[

              Container(

                  alignment: Alignment.topCenter,

                  height: 200,

                  decoration: BoxDecoration(

                    image: DecorationImage(

                      image: AssetImage("gambar/motor\_pump.jpg"),

                      fit: BoxFit.fitWidth,

                    ),

                  ),

                  child: Row(

                    mainAxisAlignment: MainAxisAlignment.spaceBetween,

                    children: <Widget>[

                      Container(

                        color: warna,

                        width: 120.0,

                        height: 60.0,

                        child: Column(

                          children: <Widget>[

                            Text(

                              "Motor Temp",

                              style: TextStyle(fontSize: 15.0),

                            ),

                            Text(

                              "${suhu.toStringAsFixed(2)}\u00B0C",

                              style: TextStyle(fontSize: 22.0),

                            )

                          ],

                        ),

                      ),

                      Container(

                        color: Colors.white60,

                        width: 120.0,

                        height: 60.0,

                        child: Column(

                          children: <Widget>[

                            Text(

                              "Water Flow",

                              style: TextStyle(fontSize: 15.0),

                            ),

                            Text(

                              "${flow.toStringAsFixed(2)}mL/s",

                              style: TextStyle(fontSize: 21.0),

                            )

                          ],

                        ),

                      )

                    ],

                  )),

              Container(

                padding: EdgeInsets.only(left: 20.0, right: 20.0, bottom: 20.0),

                child: Form(

                  key: \_formKey,

                  child: Column(

                    children: <Widget>[

                      Container(

                        height: 20,

                      ),

                      Row(

                        children: <Widget>[

                          Flexible(

                            child: TextFormField(

                              textAlign: TextAlign.center,

                              controller: sumurMin,

                              decoration: InputDecoration(

                                  hintText: "00 ~ 13",

                                  labelText: "Sumur alarm",

                                  border: OutlineInputBorder()),

                              keyboardType: TextInputType.number,

                              validator: (value) {

                                if (value.isEmpty) {

                                  return 'Tidak boleh kosong';

                                }

                                if (int.parse(value) > 13) {

                                  return 'Angka Terlalu Besar';

                                }

                                return null;

                              },

                            ),

                          ),

                          Container(

                            width: 20.0,

                          ),

                          Flexible(

                            child: TextFormField(

                              textAlign: TextAlign.center,

                              controller: sumurMax,

                              decoration: InputDecoration(

                                  hintText: "00 ~ 13",

                                  labelText: "Sumur reset",

                                  border: OutlineInputBorder()),

                              keyboardType: TextInputType.number,

                              validator: (value) {

                                if (value.isEmpty) {

                                  return 'Tidak boleh kosong';

                                }

                                if (int.parse(value) > 13) {

                                  return 'Angka Terlalu Besar';

                                }

                                return null;

                              },

                            ),

                          ),

                        ],

                      ),

                      Container(

                        height: 20,

                      ),

                      Row(

                        children: <Widget>[

                          Flexible(

                            child: TextFormField(

                              textAlign: TextAlign.center,

                              controller: tangkiMin,

                              decoration: InputDecoration(

                                  hintText: "00 ~ 13",

                                  labelText: "Tangki Min",

                                  border: OutlineInputBorder()),

                              keyboardType: TextInputType.number,

                              validator: (value) {

                                if (value.isEmpty) {

                                  return 'Tidak boleh kosong';

                                }

                                if (int.parse(value) > 13) {

                                  return 'Angka Terlalu Besar';

                                }

                                return null;

                              },

                            ),

                          ),

                          Container(

                            width: 20.0,

                          ),

                          Flexible(

                            child: TextFormField(

                              textAlign: TextAlign.center,

                              controller: tangkiMax,

                              decoration: InputDecoration(

                                  hintText: "00 ~ 13",

                                  labelText: "Tangki Max",

                                  border: OutlineInputBorder()),

                              keyboardType: TextInputType.number,

                              validator: (value) {

                                if (value.isEmpty) {

                                  return 'Tidak boleh kosong';

                                }

                                if (int.parse(value) > 13) {

                                  return 'Angka Terlalu Besar';

                                }

                                return null;

                              },

                            ),

                          ),

                        ],

                      ),

                      Container(

                        height: 20,

                      ),

                      Container(

                        width: 150,

                        height: 60,

                        child: RaisedButton.icon(

                          onPressed: () {

                            print('Button Clicked.');

                            if (\_formKey.currentState.validate()) {

                              // If the form is valid, display a Snackbar.

                              Scaffold.of(context).showSnackBar(

                                  SnackBar(content: Text('Processing Data')));

                              dbRef.child("nodeGet").update({

                                "sumur\_off\_level": int.parse(sumurMin.text)

                              });

                              dbRef.child("nodeGet").update(

                                  {"sumur\_on\_level": int.parse(sumurMax.text)});

                              dbRef.child("tankGet").update(

                                  {"tangki\_max": int.parse(tangkiMax.text)});

                              dbRef.child("tankGet").update(

                                  {"tangki\_min": int.parse(tangkiMin.text)});

                            }

                          },

                          shape: RoundedRectangleBorder(

                              borderRadius:

                                  BorderRadius.all(Radius.circular(10.0))),

                          label: Text(

                            'Simpan',

                            style: TextStyle(color: Colors.white),

                          ),

                          icon: Icon(

                            Icons.save,

                            color: Colors.white,

                          ),

                          textColor: Colors.white,

                          splashColor: Colors.red,

                          color: Colors.green,

                        ),

                      ),

                      Container(

                        height: 100,

                      ),

                    ],

                  ),

                ),

              ),

            ],

          ),

          // ),

        ),

      ),

    );

  }

}