DAFTAR PUSTAKA

Arifin, D. N., Wibisono, W., & Pratomo, B. A. (2013). Rancang Bangun Sistem Fall Detection untuk Pengguna Bergerak Berbasis Sensor Accelerometer dan Sensor Gyroscope pada Perangkat Mobile. *Jurnal Teknik POMITS*.

Benaissa, B., Koppen, M., & Yoshida, K. (2017). Activity and emotion recognition for elderly health monitoring. *Position paper ISASE*.

Bourke, A. K., & Lyons, G. M. (2008, December 3). A threshold-based fall-detection algorithm using. *Medical Engineering & Physics, 30*, 84-90.

Cherbumroong, S., Cang, S., Atkins, A., & Yu, H. (2013). Elderly activities recognition and classification for applications in assisted living. *Expert Systems with Applications, 40*, 1662-1674.

Faussett, L. (1994). *Fundamental of Neural Network (Archetectures, Algorithms, and Applications.* Upper Saddle River, New Jersey: Prentice-Hall.

Ginanto, N. (2012, November 14). *Backpropagation*. Retrieved May 27, 2018, from novikaginANTO site: https://novikaginanto.wordpress.com/2012/11/14/backpropogation/

Haykin, S. (2009). *Neural Networks and Learning Machine* (3rd ed.). (M. J. Horton, A. Dworkin, D. Mars, W. Opaluch, S. Disanno, & G. Dulles, Eds.) New Jersey: Pearson Education.

Kusumadewi, F. (2014). *Peramalan Harga Emas Menggunakan Feedforward Neural Network dengan Algoritma Backpropagation.* Yogyakarta: FMIPA UNY.

Kwapisz, J. R., Weiss, G. M., & Moore, S. A. (2010). Activity Recognition using Cell Phone Accelerometers. *SIGKDD Explorations, 12*(2), 74.

Li, Q., Stankovic, J. A., Hanson, M. A., Barth, A. T., & Zhou, G. (2009). Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-erived Posture Information. *Body Sensor Network*.

Zhou, Q., Wang, Y., & Jin, Q. (2016). SmartFDS: Design and nplementation of Falling Detection Systems on Smartphones. *IEEE*

International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom), 402-405.

Liandana, M., Mustika, I. W., & Selo. (2014, March 15). PENGEMBANGAN SISTEM DETEKSI JATUH PADA LANJUT USIA. *Seminar Nasional Teknologi Informasi dan Komunikasi*.

Mednieks, Z., Docnin, L., Meike, G. B., & Nakamura, M. (2012). *Programming Android* (2nd ed.). (A. Oram, & R. Roumeliotis, Eds.) California, United States of America: O'Reilly Media.

Olsson, F., Kok, M., Halvorsen, K., & Schon, T. B. (n.d.). Accelerometer Calibration Using Sensor Fusion with A Gyroscope.

Puspitaningrum, D. (2006). *Pengantar Jaringan Syaraf Tiruan.* Yogyakarta: Penerbit Andi.

Rakhman, A. Z., Nugroho, L. E., Widyawan, & Kurnianingsih. (2014). Fall Detection System Using Accelerometer and. *1st International Conference on Infonnation Technology, Computer and Electrical Engineering*.

Riadi, M. (2016, November 19). *Kajian Pustaka*. Retrieved May 2, 2018, from Jaringan Saraf Tiruan (JST): https://www.kajianpustaka.com/2016/11/jaringan-saraf-tiruan-jst.html

Sengto, M. A., & Leauhatong, D. (2012). Human Falling Detection Algorithm Using Back. *Biomedical Engineering International Conference*.

Siang, J. (2004). *Aplikasi Jaringan Syaraf Tiruan dan Pemrograman Menggunakan MATLAB*. Yogyakarta: Penerbit Andi.

Svozil, D., Kvasnicka, V., & Pospichal, J. (1997, June 6). Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems 39*, 43-62.

Wibisono, W., Arifin, D. N., Pratomo, B. A., Ahmad, T., & Ijtihadie, R. M. (2013). Falls Detection and Notification System Using Tri-Axial Accelerometer and Gyroscope Sensors of A Smartphone. *Conference on Technologies and Applications of Artificial Intelligence*.

N., & Sittiwanchai, T. (2017). The Development of Artificial Neural etworks (ANN) for Falls Detection. *3rd International Conference on ontrol, Automation and Robotics*, 548-550.

Zhang, S., Li, H., McCullagh, P., Nugent, C., & Zheng, H. (2013). A Real-time Falls Detection System for Elderly. *5th Computer Science and Electronic Engineering Conference*.

LAMPIRAN

Lampiran 1 Median Smoothing Filter

```
import org.apache.commons.math3.stat.StatUtils;

import java.util.LinkedList;
import java.util.List;

public class MedianSmoothingFilter {
    private static final String tag =
MedianSmoothingFilter.class.getSimpleName();

    private float timeConstant = 1;
    private float startTime = 0;
    private float timestamp = 0;
    private float hz = 0;
int count = 0;

    private int filterWindow = 3;

    private boolean dataInit;

    private LinkedList<Float> dataLists1, dataLists2, dataLists3;

    public MedianSmoothingFilter()
    {
        dataLists1 = new LinkedList<Float>();
        dataLists2 = new LinkedList<Float>();
        dataLists3 = new LinkedList<Float>();
        dataInit = false;
    }

    public void setTimeConstant(float timeConstant)
    {
        this.timeConstant = timeConstant;
    }

    public void reset()
    {
        startTime = 0;
        timestamp = 0;
        count = 0;
        hz = 0;
    }


    public float[] addSamples(float data1, float data2, float data3,
float time)
    {

        // Initialize the start time.

        dataLists1.addLast(data1);
        dataLists2.addLast(data2);
        dataLists3.addLast(data3);

        if (dataLists1.size() > filterWindow) {
            dataLists1.removeFirst();
            dataLists2.removeFirst();
            dataLists3.removeFirst();
        }

        float[] medians = new float[dataLists1.size()];
```

```
        medians[0] = (float) getMedian(dataLists1);
        medians[1] = (float) getMedian(dataLists2);
        medians[2] = (float) getMedian(dataLists3);

        //startTime = timestamp;
        return medians;
    }

    public int getFilterWindow() {

        return filterWindow;
    }

    private float getMedian(List<Float> data)
    {
        double[] values = new double[data.size()];

        for (int i = 0; i < values.length; i++)
        {
            values[i] = data.get(i).floatValue();
        }

        //return (float) StatUtils.percentile(values, 50);
        return (float) StatUtils.percentile(values, 50);
    }
}
```

Lampiran 2 Pelabelan Algoritma Threshold

```
public class ThresholdAlgorithm {
    public float alphathreshold, omegathreshold, lyingthreshold,
magnitudethreshold;

    public ThresholdAlgorithm(float alphathreshold, float omegathreshold,
float lyingthreshold, float magnitudethreshold) {
        this.alphathreshold = alphathreshold;
        this.omegathreshold = omegathreshold;
        this.lyingthreshold = lyingthreshold;
        this.magnitudethreshold = magnitudethreshold;
    }

    public boolean fallDetection (float a, float b, float c, float d){
        boolean isFallDetected = false;
        if (a > alphathreshold && b > omegathreshold){
            if (c > lyingthreshold){
                if(d < magnitudethreshold){
                    isFallDetected = true;
                }
            }
        }

        return isFallDetected;
    }
```

n 3 Segmentasi

```
ava.util.LinkedList;
```

```
public class SlidingWindowAccGyro {
    int Size;
    LinkedList<Float> AccX, AccY, AccZ;
    LinkedList<Float> GyroX, GyroY, GyroZ;

    public SlidingWindowAccGyro(int size) {
        this.Size = size;
        AccX = new LinkedList<Float>();
        AccY = new LinkedList<Float>();
        AccZ = new LinkedList<Float>();
        GyroX = new LinkedList<Float>();
        GyroY = new LinkedList<Float>();
        GyroZ = new LinkedList<Float>();
    }


    public void addAcc(float [] value){
        float value1 = value[0];
        float value2 = value[1];
        float value3 = value[2];
        if (!isFull()){
            if(!isFullAcc()) {
                AccX.add(new Float(value1));
                AccY.add(new Float(value2));
                AccZ.add(new Float(value3));
            }
            //add value
        } else {
            for(int i=0; i<Size/2; i++) {
                AccX.removeFirst();
                AccY.removeFirst();
                AccZ.removeFirst();
            }

            AccX.add(value1);
            AccY.add(value2);
            AccZ.add(value3);
        }
    }

    public void addGyro(float value[]){
        float value1 = value[0];
        float value2 = value[1];
        float value3 = value[2];
        if (!isFull()){
            if (!isFullGyro()) {
                GyroX.add(new Float(value1));
                GyroY.add(new Float(value2));
                GyroZ.add(new Float(value3));
            }
            //add value
        } else {
            for(int i=0; i<Size/2; i++) {
                GyroX.removeFirst();
                GyroY.removeFirst();
                GyroZ.removeFirst();
            }

            GyroX.add(value1);
            GyroY.add(value2);
            GyroZ.add(value3);
    }
```

```
    public Boolean isFull(){
        return (isFullAcc() && isFullGyro());
    }

    public Boolean isFullAcc(){
        return (AccX.size() >= Size && AccY.size() >= Size && AccZ.size()
>= Size);
    }

    public Boolean isFullGyro(){
        return ((GyroX.size() >= Size&& GyroY.size() >=Size &&
GyroZ.size() >= Size));
    }

    public LinkedList<Float> getAccX() {
        return AccX;
    }

    public LinkedList<Float> getAccY() {
        return AccY;
    }

    public LinkedList<Float> getAccZ() {
        return AccZ;
    }

    public LinkedList<Float> getGyroX() {
        return GyroX;
    }

    public LinkedList<Float> getGyroY() {
        return GyroY;
    }

    public LinkedList<Float> getGyroZ() {
        return GyroZ;
    }
}
```

Lampiran 4 Fitur Ektraksi

```
import java.util.Collections;
import java.util.LinkedList;

import static java.lang.Float.NaN;

public class HalfSlidingWindow {
    private int Size;
    private LinkedList<Float> samplesAcc, samplesGyro, samplesTheta;
    float alphasumbuY, deltaAcc, deltaGyro;
    Float Alpha, Omega, Theta;
    Exception exception;

    ic HalfSlidingWindow() {
    samplesAcc = new LinkedList<Float>();
    samplesGyro = new LinkedList<Float>();
    samplesTheta = new LinkedList<Float>();
```

```
    public void setSize(int size) {
        this.Size = size;
    }

    public int getSize() {
        return Size;
    }

    public void add(float Avalue1, float Avalue2, float Avalue3,
                    float Gvalue1, float Gvalue2, float Gvalue3){
        try{
            Alpha = AlphaMagnitude(Avalue1, Avalue2,Avalue3 );
            Omega = OmegaMagnitude(Gvalue1, Gvalue2, Gvalue3);
            Theta = Theta(Avalue2);
        }catch(Exception e){
            exception = e;
        }

        if (!isFull()){
            //samplesAcc.add(new Float(Alpha));
            //samplesGyro.add(new Float(Omega));
            //samplesTheta.add(new Float(Theta));
            samplesAcc.add(Alpha);
            samplesGyro.add(Omega);
            samplesTheta.add(Theta);
            //add value
        } else {
            for(int i=0; i<Size; i++) {
                samplesAcc.removeFirst();
                samplesGyro.removeFirst();
                samplesTheta.removeFirst();
            }
            samplesAcc.add(Alpha);
            samplesGyro.add(Omega);
            samplesTheta.add(Theta);
        }
    }

    public float AlphaMagnitude (float value1, float value2, float
value3){
        float alpha = (float) Math.sqrt(Math.pow(value1,
2)+Math.pow(value2,2)+Math.pow(value3,2));
        return alpha;
    }

    public float OmegaMagnitude (float value1, float value2, float
value3){
        float omega = (float)
Math.sqrt(Math.pow(value1,2)+Math.pow(value2,2)+Math.pow(value3,2));
        return omega;
    }

    public float Theta(float Y){
        float theta = (float) ((Math.acos(Y) / 9.8) * 180);

        if(theta == NaN){
            theta = (float) 0;
        }

        return theta;


    ic void clear(){
```

```
            samplesAcc.clear();
            samplesGyro.clear();
        }

        public Boolean isFull(){
            return isAccFull() && isGyroFull() && isThetaFull();
        }

        public Boolean isAccFull(){
            return (samplesAcc.size() >= Size);
        }

        public Boolean isGyroFull(){
            return (samplesGyro.size() >= Size);
        }

        public Boolean isThetaFull(){
            return (samplesTheta.size() >= Size);
        }

        public Float maxAcc(){
            Float maxacc = Collections.max(samplesAcc);
            return maxacc;
        }
        public Float minAcc(){
            Float minacc = Collections.min(samplesAcc);
            return minacc;
        }

        public Float maxGyro(){
            Float maxgyro = Collections.max(samplesGyro);
            return maxgyro;
        }

        public Float minGyro(){
            Float mingyro = Collections.min(samplesGyro);
            return mingyro;
        }

        public Float max(LinkedList<Float> samples){
            return Collections.max(samples);
        }

        public Float min(LinkedList<Float> samples){
            return Collections.min(samples);
        }

        //public void setDelta(LinkedList<Float> samplesA, LinkedList<Float>
    samplesB){
        public void setDelta(){


        }

        public Float getDeltaAcc(){
            deltaAcc = max(samplesAcc)-min(samplesAcc);
            return deltaAcc;


            ic Float getDeltaGyro(){
            deltaGyro = max(samplesGyro)- min(samplesGyro);
            return deltaGyro;
```

```
    public LinkedList<Float> getInputAcc() {
        return samplesAcc;
    }

    public LinkedList<Float> getInputGyro() {
        return samplesGyro;
    }

    public LinkedList<Float> getInputTheta() {
        return samplesTheta;
    }

}
```

Lampiran 5 Database

```
import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

DatabaseReference blewah = database.getReference("world");

private void writeInput(float sampleA, float sampleA1, float sampleA2,
float sampleA3,
                        float samplesB, float sampleB1, float sampleB2,
float sampleB3, float samplesC,
                         int tanda, int i,int kali, float timestamp,
boolean fallDetected, int batch) {
    boolean label = fallDetected;
    //float Timestamp = timest;
    HashMap<String, Object> result1 = new HashMap<>();


        result1.put("Acc X", sampleA1);
        result1.put("Acc Y", sampleA2);
        result1.put("Acc Z", sampleA3);
        result1.put("Acc ", sampleA);

        result1.put("Gyro X", sampleB1);
        result1.put("Gyro Y", sampleB2);
        result1.put("Gyro Z", sampleB3);
        result1.put("Gyro ", samplesB);

        result1.put("Theta", samplesC);
        result1.put("label", label);
        result1.put("i", i);
        result1.put("kali", kali+1);
        result1.put("waktu", timestamp);



    Map<String, Object> result = result1;

blewah.child("kodong2").child(String.valueOf(batch)).child(String.valueOf
(tanda+1)).updateChildren(result)
        // ori.child("tempo").updateChildren(result)
        .addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                Toast.makeText(coba2.this, "masuk",
NGTH_SHORT).show();
```

```
                }
            })
            .addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    Toast.makeText(coba2.this, "nggak",
Toast.LENGTH_SHORT).show();
                    Log.d(TAG, e.toString());
                }
            });

}
@Override
public void onSensorChanged(SensorEvent sensorEvent) {
…
    writeInput(Window.getInputAcc().get(i),
windowAccGyro.getAccX().get(i),
        windowAccGyro.getAccY().get(i), windowAccGyro.getAccZ().get(i),
        Window.getInputGyro().get(i), windowAccGyro.getGyroX().get(i),
        windowAccGyro.getGyroY().get(i), windowAccGyro.getGyroZ().get(i),
        Window.getInputTheta().get(i), tanda,  i, lap,time, fallDetected,
batch);
…
}
```

Lampiran 6 Aplikasi Deteksi Jatuh

```
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;

import android.support.annotation.NonNull;
import android.support.v4.media.MediaMetadataCompat;
import android.support.v7.app.AppCompatActivity;

import android.util.EventLog;
import android.util.Log;
import android.view.View;
import android.view.ViewDebug;
import android.view.Window;
import android.view.WindowAnimationFrameStats;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

ava.sql.Timestamp;
ava.util.HashMap;
ava.util.LinkedList;
ava.util.Map;
ava.util.Queue;
```

```
import static java.lang.Float.NaN;


public class coba2 extends AppCompatActivity implements
SensorEventListener {
    private int kali = 0, kaliTotal, iterasi = 0, lap = 0, batch
=1,tanda=0,polll;
    private static final String TAG = coba2.class.getName();
    private int hiddenlayerN =50, inputN;
    float WeightsInput[][] =new float[inputN][hiddenlayerN];
    float BiasInput[][] =new float[1][hiddenlayerN];
    float z_in[] =new float[hiddenlayerN];
    float z[] =new float[hiddenlayerN];
    float sumInput= (float) 0;
    float sumLayer = (float) 0;

    float WeightsLayer[][] =new float[hiddenlayerN][1];
    float BiasLayer[][] =new float[1][1];
    float y_in[] =new float[1];
    float y[] =new float[1];

    private SensorManager mSensor;
    private Sensor sensorA, sensorB;

    float TimestampAwal, Timestamp, TimestampSetAwal, time;
    private static final float NS2S = 1.0f / 1000000000.0f;
    int sample = 0;

    private final float alpha_THRESHOLD = (float) 1.7753;
    private final float omega_THRESHOLD = (float) 0.8579;
    private final float lying_THRESHOLD = (float) 30;
    private final float magnitude_THRESHOLD = (float) 0.5;



    private float alphaMagnitude, lastalphaMagnitude, minalphaMagnitude,
deltaalphaMagnitude;
    private float omegaMagnitude, lastomegaMagnitude,
deltaomegaMagnitude;
    private float Alpha, Omega, Theta, Minalpha, Waktu;
    private double waktu;
    //private int waktu;
    private float thetaMagnitude;
    float gravity[] = new float[3];
    float linear_acceleration[] = new float[3];
    float angular_acceleration[] = new float[3];
    private Boolean firstChange = false, mInitialized = false,
fallDetected, adaPass = false;
    private String valueUsername;

    private Button Button1, Button2;
    private TextView kaliTV, TV1, TV2, TV3;
    private int windowSize = 20;
    private boolean isOFF=false, isNotAllowed = false, init=false;
    private Samee samee = new Samee();
    float timeConstant = (float) 2.56;
    ate HalfSlidingWindow Window = new HalfSlidingWindow();
    ate MedianSmoothingFilter medianFilterAcc = new
    othingFilter();
    ate MedianSmoothingFilter medianFilterGyro = new
    othingFilter();
    ate SlidingWindowAccGyro windowAccGyro = new
    indowAccGyro(windowSize);
```

```
    private SlidingWindowAcc windowAcc = new
SlidingWindowAcc(windowSize);
    private SlidingWindowGyro windowGyro = new
SlidingWindowGyro(windowSize);
    private simpan simpann = new simpan(windowSize);
    ThresholdAlgorithm thresholdAlgorithm = new
ThresholdAlgorithm(alpha_THRESHOLD, omega_THRESHOLD,
            lying_THRESHOLD, magnitude_THRESHOLD);
    //private EditText editUsername;


    FirebaseDatabase database = FirebaseDatabase.getInstance();
    DatabaseReference myRef = database.getReference("message");
    DatabaseReference clothes = database.getReference("air");
    DatabaseReference ori = database.getReference("clothes");
    DatabaseReference blewah = database.getReference("world");

    /*private FirebaseFirestore db = FirebaseFirestore.getInstance();
    private CollectionReference fallDetectionRef = db.collection("Fall
Detection");
    private DocumentReference fallRef =
fallDetectionRef.document("Fall");
    private DocumentReference nonfallRef =
fallDetectionRef.document("Non-Fall");*/

    //private CollectionReference waktu = db.collection("Sensor");

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_coba2);

        mSensor = (SensorManager) getSystemService(SENSOR_SERVICE);
        assert mSensor != null;
        sensorA = mSensor.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
        sensorB = mSensor.getDefaultSensor(Sensor.TYPE_GYROSCOPE);
        Button1 = findViewById(R.id.BUTTON1);
        Button2 = findViewById(R.id.BUTTON2);
        kaliTV = findViewById(R.id.KALI);
        //editUsername = findViewById(R.id.EditUsername);

        if (getIntent().getExtras() != null) {
            //if(getIntent().getStringExtra("valueUsername")!=null){
            Bundle bundle = getIntent().getExtras();
            valueUsername = bundle.getString("valueUsername");
            //valueUsername = getIntent().getStringExtra(valueUsername);
            if (valueUsername != null) {
                adaPass = true;
                valueUsername = "anitaa";
            }
        }

        TV1 = findViewById(R.id.tv1);
        TV2 = findViewById(R.id.tv2);
        TV3 = findViewById(R.id.tv3);
        if (sensorB != null) {
            // Success!
            Toast.makeText(coba2.this, "nyalaji sensornya",
NGTH_SHORT).show();
        } else {
            // Failure! No pressure sensor.
            Toast.makeText(coba2.this, "errorki anu",
NGTH_SHORT).show();
```

```
        }
        if(!isOFF){
            mSensor.registerListener(coba2.this, sensorA,
SensorManager.SENSOR_DELAY_NORMAL);
            mSensor.registerListener(coba2.this, sensorB,
SensorManager.SENSOR_DELAY_NORMAL);
        }

        //mSensor.registerListener(coba2.this, sensorA, 100000);
        //mSensor.registerListener(coba2.this, sensorB, 5000000);

        /*if(valueUsername.isEmpty()){
            onPause();
        }*/

        /*ValueEventListener dsListener = new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                // Get Post object and use the values to update the UI
                //datasensor ds =
dataSnapshot.getValue(datasensor.class);
                showData(dataSnapshot);
                // ...
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {
                // Getting Post failed, log a message
                Log.w(TAG, "loadDS:onCancelled",
databaseError.toException());
                // ...
            }
        };*/

    }

    private float AddData_Acc(float ax, float ay, float az) {
        // TODO Auto-generated method stub
        float alpha = (float) Math.sqrt(ax * ax + ay * ay + az * az);
        /*for(int i=0;i<=BUFF_SIZE-2;i++){
            window[i]=window[i+1];
        }
        window[BUFF_SIZE-1]=alphaMagnitude;*/
        return alpha;
    }



    private float Filter_Acc(float axis, int i) {
        final float alfa = (float) 0.8;


        gravity[i] = alfa * gravity[i] + (1 - alfa) * axis;
        linear_acceleration[i] = axis - gravity[i];
        return linear_acceleration[i];
    }

        ate float AddData_Gyro(double pitch, double roll, double yaw) {
        // TODO Auto-generated method stub
        float omega = (float) Math.sqrt(pitch * pitch + roll * roll + yaw

        /*for(int i=0;i<=BUFF_SIZE-2;i++){
            window[i]=window[i+1];
```

```
        }
        window[BUFF_SIZE-1]= omegaMagnitude;*/
        return omega;

    }

    private void writeInputAcc(float sampleA, float sampleB, float
sampleC, int kali, float timestamp){
        HashMap<String, Object> result1 = new HashMap<>();
        result1.put("x", sampleA);
        result1.put("y", sampleB);
        result1.put("z", sampleC);
        result1.put("waktu", timestamp);

        Map<String, Object> result = result1;
        ori.child("plot
Acc").child(String.valueOf(kali+1)).updateChildren(result)
                .addOnSuccessListener(new OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void aVoid) {
                        Toast.makeText(coba2.this, "masuk",
Toast.LENGTH_SHORT).show();
                    }
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Toast.makeText(coba2.this, "nggak",
Toast.LENGTH_SHORT).show();
                        Log.d(TAG, e.toString());
                    }
                });
    }


    //private void writeInput(LinkedList<Float> samplesA,
LinkedList<Float> samplesB, LinkedList<Float> samplesC, int kali, float
timestamp, boolean fallDetected) {
    private void writeInput(float sampleA, float sampleA1, float
sampleA2, float sampleA3,
                            float samplesB, float sampleB1, float
sampleB2, float sampleB3, float samplesC,
                             int tanda, int i,int kali, float timestamp,
boolean fallDetected, int batch) {
        boolean label = fallDetected;
        //float Timestamp = timest;
        HashMap<String, Object> result1 = new HashMap<>();
        /*Float[] acc = new Float[windowSize+1];
        Float[] gyro = new Float[windowSize+1];
        Float[] theta = new Float[windowSize+1];

        for (int i=0; i < windowSize+1; i++){
            acc[i] = samplesA.get(i);
            gyro[i] = samplesB.get(i);
            theta[i] = samplesC.get(i);
        //}


        //for (int i = 0; i < windowSize+1; i++) {
            //Log.d(TAG, "writeInput: " + i);
            //Log.d(TAG, "sample " + acc[i] + "; i = " + i + "; kali " +
);
            */
```

```
                result1.put("Acc X", sampleA1);
                result1.put("Acc Y", sampleA2);
                result1.put("Acc Z", sampleA3);
                result1.put("Acc ", sampleA);

                result1.put("Gyro X", sampleB1);
                result1.put("Gyro Y", sampleB2);
                result1.put("Gyro Z", sampleB3);
                result1.put("Gyro ", samplesB);

                result1.put("Theta", samplesC);
                result1.put("label", label);
                result1.put("i", i);
                result1.put("kali", kali+1);
                result1.put("waktu", timestamp);



//ori.child("blurp").child(String.valueOf(kali)).updateChildren(result1)

        //}

        Map<String, Object> result = result1;

blewah.child("kodong2").child(String.valueOf(batch)).child(String.valueOf
(tanda+1)).updateChildren(result)
                // ori.child("tempo").updateChildren(result)
                .addOnSuccessListener(new OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void aVoid) {
                        Toast.makeText(coba2.this, "masuk",
Toast.LENGTH_SHORT).show();
                    }
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Toast.makeText(coba2.this, "nggak",
Toast.LENGTH_SHORT).show();
                        Log.d(TAG, e.toString());
                    }
                });
        //result.clear();




    }

    /*private void writeNewData(String name, int sesi, int kali, float
alpha, float omega, float theta,
                              boolean label, double waktu) {
        //do{
            // Create new post at /user-posts/$userid/$postid and at
            // /posts/$postid simultaneously
            // String key =
ild(String.valueOf(waktu)).push().getKey();

            datasensor ds = new datasensor(name, sesi, alpha, omega,
abel, waktu);
            Map<String, Object> dsValues = ds.toMap();
```

```
            //Map<String, Object> childUpdates = new HashMap<>();
            //childUpdates.put("/posts/" + key, postValues);
            //childUpdates.put("/user-posts/" + waktu + "/" + key,
postValues);
            int time = (int) waktu*100;

clothes.child("Worth").child(String.valueOf(kali)).updateChildren(dsValue
s)
                    .addOnSuccessListener(new OnSuccessListener<Void>() {
                        @Override
                        public void onSuccess(Void aVoid) {
                            Toast.makeText(coba2.this, "succes added to
RTD", Toast.LENGTH_SHORT).show();
                        }
                    })
                    .addOnFailureListener(new OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull Exception e) {
                            Toast.makeText(coba2.this, "error adding",
Toast.LENGTH_SHORT).show();
                            Log.d(TAG, e.toString());
                        }
                    });
        //} while(valueUsername.isEmpty());
    }

    private void showData(DataSnapshot dataSnapshot){
        for(DataSnapshot dSnapshot: dataSnapshot.getChildren()){
            datasensor ds = new datasensor();

            for(int i=10; i<11;i++){

ds.setAlpha(dSnapshot.child("Anita").child("0").child(String.valueOf(i)).
getValue(datasensor.class).getAlpha());

ds.setOmega(dSnapshot.child("Anita").child("0").child(String.valueOf(i)).
getValue(datasensor.class).getOmega());

ds.setTheta(dSnapshot.child("Anita").child("0").child(String.valueOf(i)).
getValue(datasensor.class).getTheta());

ds.setWaktu(dSnapshot.child("Anita").child("0").child(String.valueOf(i)).
getValue(datasensor.class).getWaktu());
                Log.d(TAG, "showData: alpha: "+ds.getAlpha());
                Log.d(TAG, "showData: omega: "+ds.getOmega());
                Log.d(TAG, "showData: theta: "+ds.getTheta());
                Log.d(TAG, "showData: waktu: "+ds.getWaktu());

                ArrayList<Float> array = new ArrayList<>();
                array.add(ds.getAlpha());
                array.add(ds.getOmega());
                array.add(ds.getTheta());
                array.add((float)ds.getWaktu());
                ArrayAdapter adapter = new ArrayAdapter(this,
android.R.layout.simple_list_item_1, array);
                //listView.setAdapter(adapter);
            }
        }


    ic boolean StartBtn(View v) {
    //kali=0;
    //onStart();
```

```
        //startService(new Intent(coba2.this, AndroidService.class));
        onResume();
        return true;
    }

    public boolean StopBtn(View v) {
        //try{
        //onPause();
        mSensor.unregisterListener(coba2.this);
        isOFF = true;
        //}catch (Exception e) {
        //    Log.d(TAG, "StopBtn: " + e.toString());
        //}
        //stopService(new Intent(coba2.this, AndroidService.class));
        return true;
    }


    @Override
    public void onSensorChanged(SensorEvent sensorEvent) {
        try {

            //Thread.sleep(40);
            if (!firstChange) {
                TimestampAwal = sensorEvent.timestamp * NS2S;
                TimestampSetAwal = TimestampAwal;
                firstChange = true;
                Timestamp = sensorEvent.timestamp * NS2S;

                //medianFilterAcc.setTimeConstant(timeConstant);
                //medianFilterGyro.setTimeConstant(timeConstant);

                //myRef.child("contoh").child("koro").setValue("tek
retek");

                //Buttonbarua.setText(String.valueOf(TimestampAwal));
            } else {
                Timestamp = sensorEvent.timestamp * NS2S;
            }
            //aButton.setText("ff");

            try {
                switch (sensorEvent.sensor.getType()) {
                    case Sensor.TYPE_ACCELEROMETER:

//TV1.setText(String.valueOf(sensorEvent.values[0]));

//TV2.setText(String.valueOf(sensorEvent.values[1]));

//TV3.setText(String.valueOf(sensorEvent.values[2]));
                        for (int ite = 0; ite <= 2; ite++) {
                            if (sensorEvent.values[ite] == NaN) {
                                //sensorEvent.values[ite] = 0;
                            }
                            linear_acceleration[ite] =
cc(sensorEvent.values[ite], ite);
                        }
                        //alphaMagnitude =
Acc(linear_acceleration[0], linear_acceleration[1],
cceleration[2]);
                        //windowAcc.add(linear_acceleration[0],
cceleration[1], linear_acceleration[2]);
```

```
                        linear_acceleration = medianFilterAcc.addSamples
                                (linear_acceleration[0],
linear_acceleration[1], linear_acceleration[2], sensorEvent.timestamp);

//windowAccGyro.setSizeAcc(medianFilterAcc.getFilterWindow());
                        windowAccGyro.addAcc(linear_acceleration);
                        Log.d(TAG, "onSensorChanged: masuk baru ACC");
                        //Log.d(TAG, "!onSensorChanged: Accelerometer " +
linear_acceleration[0]);
                        break;
                    case Sensor.TYPE_GYROSCOPE:
                        //omegaMagnitude =
AddData_Gyro(sensorEvent.values[0], sensorEvent.values[1],
sensorEvent.values[2]);
                        //windowGyro.add(sensorEvent.values[0],
sensorEvent.values[1], sensorEvent.values[2]);
                        for (int ite = 0; ite <= 2; ite++) {
                            if (sensorEvent.values[ite] == NaN) {
                                //sensorEvent.values[ite] = 0;
                            }
                        }
                        angular_acceleration =
medianFilterGyro.addSamples
                                (sensorEvent.values[0],
sensorEvent.values[1], sensorEvent.values[2], Timestamp);


//windowAccGyro.setSizeGyro(medianFilterGyro.getFilterWindow());
                        windowAccGyro.addGyro(angular_acceleration);
                        Log.d(TAG, "onSensorChanged: masuk baru GYRO");
                        //Log.d(TAG, "!onSensorChanged: Gyroscope " +
sensorEvent.values[0]);
                        break;
                }
            }catch (Exception e){
                Log.d(TAG, "onSensorChanged: error add sensor "+
e.toString());
            }



            //Log.d(TAG, "onSensorChanged: MF Start time = "+
medianFilterAcc.startTime + " & ts = "+ medianFilterAcc.timestamp);
            //Log.d(TAG, "onSensorChanged: MF Count "+
medianFilterAcc.count);
            //Log.d(TAG, "onSensorChanged: MF hz = "+
medianFilterAcc.hz);
            Log.d(TAG, "onSensorChanged: MF window filter "+
medianFilterAcc.getFilterWindow());
            Log.d(TAG, "onSensorChanged: MF Window Size = " +
windowSize);
            //thetaMagnitude = (float)
((Math.acos(linear_acceleration[1]) / 9.8) * 180);


            //float peon[][] = new float[200][1];
            //peon[kali-1][0]= alphaMagnitude;
            //con[kali-1][1]= omegaMagnitude;
            //con[kali-1][2]= thetaMagnitude;
            //con[kali-1][3]= minalphaMagnitude;
            //con[kali-1][4] = (float) waktu;
            Log.d(TAG, "onSensorChanged: ACC -> X = " +
cGyro.getAccX());
```

```
            Log.d(TAG, "onSensorChanged: ACC -> Y = " +
windowAccGyro.getAccY());
            Log.d(TAG, "onSensorChanged: ACC -> Z = " +
windowAccGyro.getAccZ());

            //Log.d(TAG, "onSensorChanged: GYRO ->" + omegaMagnitude);
            Log.d(TAG, "onSensorChanged: GYRO -> X = " +
windowAccGyro.getGyroX());
            Log.d(TAG, "onSensorChanged: GYRO -> Y = " +
windowAccGyro.getGyroY());
            Log.d(TAG, "onSensorChanged: GYRO -> Z = " +
windowAccGyro.getGyroZ());
            int n = 100;



            Window.setSize(windowSize);
            for (int i = 0; i < windowSize; i++) {
                Log.d(TAG, "onSensorChanged: I = "+i+ " LAP = " + lap);
                Log.d(TAG, "onSensorChanged: K " +
windowAccGyro.isFullAcc()+ " and "+ windowAccGyro.isFullGyro());
                Log.d(TAG, "onSensorChanged: " +
String.valueOf(windowAccGyro.isFull()));

                Log.d(TAG, "onSensorChanged: WINDOW FULL = "+
Window.isAccFull()+ " & "+ Window.isGyroFull());
                //Log.d(TAG, "onSensorChanged: A ACC ->" +
Window.getInputAcc()+ " SIZE = "+Window.getInputAcc().size());
                //Log.d(TAG, "onSensorChanged: A GYRO ->" +
Window.getInputGyro()+ " SIZE = "+Window.getInputGyro().size());
                //Log.d(TAG, "onSensorChanged: A THETA ->" +
Window.getInputTheta()+ " SIZE = "+Window.getInputTheta().size());
                //if (windowAcc.isFull() && windowGyro.isFull()) {
                if (windowAccGyro.isFull()) {
                    //if(!Window.isFull()){
                        try {
                            //Window.add(windowAcc.getAccX().get(i),
windowAcc.getAccY().get(i), windowAcc.getAccZ().get(i),
windowGyro.getGyroX().get(i), windowGyro.getGyroY().get(i),
windowGyro.getGyroZ().get(i));
                            Window.add(windowAccGyro.getAccX().get(i),
windowAccGyro.getAccY().get(i),
                                    windowAccGyro.getAccZ().get(i),
windowAccGyro.getGyroX().get(i),
                                    windowAccGyro.getGyroY().get(i),
windowAccGyro.getGyroZ().get(i));
                            //Window.add((float) 0.663486123,  (float)
0.879630566, (float)1.248848796, (float)-0.33114624,   (float)-
0.147689819, (float)-0.16784668 );

                            Log.d(TAG, "onSensorChanged: window Alpha = "
+ Window.Alpha);

                        } catch (Exception e) {
                            Log.d(TAG, "onSensorChanged: window add-> " +
e.getMessage());
                        }
                    //}
                    Log.d(TAG, "onSensorChanged: ERROR ->
.exception);
                }
                Log.d(TAG, "onSensorChanged: A ACC ->" +
etInputAcc()+ " SIZE = "+Window.getInputAcc().size());
```

```
                Log.d(TAG, "onSensorChanged: A GYRO ->" +
Window.getInputGyro()+ " SIZE = "+Window.getInputGyro().size());
                Log.d(TAG, "onSensorChanged: A THETA ->" +
Window.getInputTheta()+ " SIZE = "+Window.getInputTheta().size());
                Log.d(TAG, "onSensorChanged: " +
String.valueOf(windowAcc.isFull()) +" & " +
String.valueOf(windowGyro.isFull()));

            }

            /*
            //z_in = (float) 0;
            if(Window.isFull()&&windowAccGyro.isFull()){
                for (int o = 0; o<windowSize; o++){
                    for(int q = 0; q < hiddenlayerN; q++){
                        z_in[q] +=
WeightsInput[0][q]*Window.getInputAcc().get(o);
                        z_in[q] +=
WeightsInput[1][q]*windowAccGyro.getAccX().get(o);
                        z_in[q] +=
WeightsInput[2][q]*windowAccGyro.getAccY().get(o);
                        z_in[q] +=
WeightsInput[3][q]*windowAccGyro.getAccZ().get(o);
                        z_in[q] +=
WeightsInput[4][q]*Window.getInputGyro().get(o);
                        z_in[q] +=
WeightsInput[5][q]*windowAccGyro.getGyroX().get(o);
                        z_in[q] +=
WeightsInput[6][q]*windowAccGyro.getGyroY().get(o);
                        z_in[q] +=
WeightsInput[7][q]*windowAccGyro.getGyroZ().get(o);
                        z_in[q] +=
WeightsInput[8][q]*Window.getInputTheta().get(o);
                        z_in[q] += BiasInput[q][0];
                    }


                    for(int v = 0; v < hiddenlayerN; v++){
                        y_in[0] += WeightsLayer[v][0]*z[v];

                    }
                    y_in[0] += BiasLayer[0][0];
                    y[0] = y_in[0];

                    if(y[0]==1){
                        fallDetected = true;
                    }else{
                        fallDetected = false;
                    }

                }
            }*/


            Log.d(TAG, "onSensorChanged: init "+ init);
            if(!init){
                if(windowAccGyro.isFull()){
                    Float a = Window.getDeltaAcc();
                    Float b = Window.getDeltaGyro();

                    //TV1.setText(String.valueOf(a));
                    //TV2.setText(String.valueOf(Window.minAcc()));
```

```
                    //fallDetected = thresholdAlgorithm.fallDetection(a,
b, thetaMagnitude, Window.minAcc());
                    Log.d(TAG, "INPUT AWAL -> "+
System.currentTimeMillis());
                    time = Timestamp - TimestampAwal;
                    try{
                        for (int i=0; i<windowSize; i++) {
                            fallDetected =
thresholdAlgorithm.fallDetection(a, b, Window.getInputTheta().get(i),
Window.minAcc());
                            //fallDetected =
thresholdAlgorithm.fallDetection((float)3, (float)2.2, 30,
(float)0.0001);

//writeInputAcc(windowAccGyro.getAccX().get(i),
windowAccGyro.getAccY().get(i), windowAccGyro.getAccZ().get(i), lap,
time);
                            //writeInput(Window.getInputAcc().get(i),
Window.getInputGyro().get(i), Window.getInputTheta().get(i), lap, i,
Timestamp, fallDetected);
                            writeInput(Window.getInputAcc().get(i),
windowAccGyro.getAccX().get(i),
                                    windowAccGyro.getAccY().get(i),
windowAccGyro.getAccZ().get(i),
                                    Window.getInputGyro().get(i),
windowAccGyro.getGyroX().get(i),
                                    windowAccGyro.getGyroY().get(i),
windowAccGyro.getGyroZ().get(i),
                                    Window.getInputTheta().get(i), tanda,
i, lap,time, fallDetected, batch);

                            lap++;
                            tanda++;
                            Log.d(TAG, "onSensorChanged: INPUT I = "+i);
                            Log.d(TAG, "onSensorChanged: batch -> FALL =
"+fallDetected);
                            polll = i;
                        }


                        /*for (int i=0; i<windowSize; i++) {

simpann.setSimpan(Window.getInputAcc().get(i),
windowAccGyro.getAccX().get(i),
                                    windowAccGyro.getAccY().get(i),
windowAccGyro.getAccZ().get(i),
                                    Window.getInputGyro().get(i),
windowAccGyro.getGyroX().get(i),
                                    windowAccGyro.getGyroY().get(i),
windowAccGyro.getGyroZ().get(i),
                                    Window.getInputTheta().get(i), i);
                        }*/
                        Log.d(TAG, "onSensorChanged: INPUT SUCCESS !
batch = "+batch);
                    }catch (Exception e){
                        Log.d(TAG, "INPUT ERROR -> "+e.toString());
                    }
                    if(lap%2000==0){
                        batch++;
                        tanda=1;
                        Log.d(TAG, "onSensorChanged: batch++");

                    }
```

```
                    Log.d(TAG, "INPUT AKHIR ->
"+System.currentTimeMillis());
                    Log.d(TAG, "onSensorChanged: batch = "+batch+"
(init)"+", lap = "+lap+", tanda = "+tanda+ ", i = "+polll+", SIZE =
"+Window.getSize());
                    TV3.setText(String.valueOf(a) + " and " +
String.valueOf(b) + " serta " + Window.minAcc());
                }

                init = true;
            } else{
                /*
                if(windowAccGyro.isFull()){
                    for(int i=0; i<windowSize; i++){


if((Window.getInputAcc().get(i)==simpann.getSimpan(0, i)) ||
(windowAccGyro.getAccX().get(i)==simpann.getSimpan(1, i)) ||

(windowAccGyro.getAccY().get(i)==simpann.getSimpan(2, i)) ||
(windowAccGyro.getAccZ().get(i)==simpann.getSimpan(3, i))||

(Window.getInputGyro().get(i)==simpann.getSimpan(4, i)) ||
(windowAccGyro.getGyroX().get(i)==simpann.getSimpan(5, i)) ||

(windowAccGyro.getGyroY().get(i)==simpann.getSimpan(6, i)) ||
(windowAccGyro.getGyroZ().get(i)==simpann.getSimpan(7, i)))
                            isNotAllowed = true;
                    }
                }*/

                Log.d(TAG, "onSensorChanged: K " +
windowAccGyro.isFullAcc()+ " and "+ windowAccGyro.isFullGyro());
                Log.d(TAG, "onSensorChanged: isnotallowed = "+
isNotAllowed);
                if(windowAccGyro.isFull()&&(!isNotAllowed)){
                    Float a = Window.getDeltaAcc();
                    Float b = Window.getDeltaGyro();

                    //TV1.setText(String.valueOf(a));
                    //TV2.setText(String.valueOf(Window.minAcc()));
                    //fallDetected = thresholdAlgorithm.fallDetection(a,
b, thetaMagnitude, Window.minAcc());
                    Log.d(TAG, "INPUT AWAL -> "+
System.currentTimeMillis());
                    time = Timestamp - TimestampAwal;
                    try{
                        for (int i=0; i<windowSize; i++) {
                            fallDetected =
thresholdAlgorithm.fallDetection(a, b, Window.getInputTheta().get(i),
Window.minAcc());
                            //fallDetected =
thresholdAlgorithm.fallDetection((float)3, (float)2.2, 30,
(float)0.0001);
                            //Log.d(TAG, "onSensorChanged: batch -> FALL
= "+fallDetected);

            nputAcc(windowAccGyro.getAccX().get(i),
            cGyro.getAccY().get(i), windowAccGyro.getAccZ().get(i), lap,

                            //writeInput(Window.getInputAcc().get(i),
            etInputGyro().get(i), Window.getInputTheta().get(i), lap, i,
            o, fallDetected);
```

```
                                /*writeInput(Window.getInputAcc().get(i),
windowAccGyro.getAccX().get(i),
                                        windowAccGyro.getAccY().get(i),
windowAccGyro.getAccZ().get(i),
                                        Window.getInputGyro().get(i),
windowAccGyro.getGyroX().get(i),
                                        windowAccGyro.getGyroY().get(i),
windowAccGyro.getGyroZ().get(i),
                                        Window.getInputTheta().get(i), tanda,
i, lap, time, fallDetected, batch);*/

                                lap++;
                                tanda++;
                                polll = i;
                                Log.d(TAG, "onSensorChanged: INPUT I = "+i+
",lap = "+lap);
                            }

                            /*
                            for (int i=0; i<windowSize; i++){

simpann.getSimpan(Window.getInputAcc().get(i),
windowAccGyro.getAccX().get(i),
                                        windowAccGyro.getAccY().get(i),
windowAccGyro.getAccZ().get(i),
                                        Window.getInputGyro().get(i),
windowAccGyro.getGyroX().get(i),
                                        windowAccGyro.getGyroY().get(i),
windowAccGyro.getGyroZ().get(i),
                                        Window.getInputTheta().get(i), i );
                            }*/


                                Log.d(TAG, "onSensorChanged: INPUT SUCCESS !
batch = "+batch);

                        }catch (Exception e){
                            Log.d(TAG, "INPUT ERROR -> "+e.toString());
                        }
                        if(lap%2000==0&&lap!=0){
                            batch++;
                            tanda=0;
                            Log.d(TAG, "onSensorChanged: batch++");
                        }
                        Log.d(TAG, "INPUT AKHIR ->
"+System.currentTimeMillis());
                        Log.d(TAG, "onSensorChanged: batch = "+batch+ ", lap
= "+lap+ ", tanda = "+tanda+ ", i = "+polll+", SIZE =
"+Window.getSize());
                        TV3.setText(String.valueOf(a) + " and " +
String.valueOf(b) + " serta " + Window.minAcc());


                        //}
                    }
                }

                //Log.d(TAG, "onSensorChanged: B isnotallowed = "+
owed + ", lap = "+lap + " simpan[0][0] = "+ simpann.getSimpan(0,0

                String blow = "" ;
                TV1.setText(String.valueOf(Window.isFull()));
```

```
        //Log.d(TAG, "onSensorChanged: ACC ->" + alphaMagnitude);
            /*
            Log.d(TAG, "onSensorChanged: ACC -> X = " +
windowAccGyro.getAccX());
            Log.d(TAG, "onSensorChanged: ACC -> Y = " +
windowAccGyro.getAccY());
            Log.d(TAG, "onSensorChanged: ACC -> Z = " +
windowAccGyro.getAccZ());

                    //Log.d(TAG, "onSensorChanged: GYRO ->" +
omegaMagnitude);
            Log.d(TAG, "onSensorChanged: GYRO -> X = " +
windowAccGyro.getGyroX());
            Log.d(TAG, "onSensorChanged: GYRO -> Y = " +
windowAccGyro.getGyroY());
            Log.d(TAG, "onSensorChanged: GYRO -> Z = " +
windowAccGyro.getGyroZ());
            */

                    //Log.d(TAG, "onSensorChanged: THETA ->" +
thetaMagnitude);

            //Window.getInput()
            Button1.setText(String.valueOf(TimestampAwal));
            Button2.setText(String.valueOf(Timestamp));
            kaliTV.setText(String.valueOf(TimestampSetAwal));

            //waktu = Timestamp;

            /*if (Timestamp >= TimestampAwal + 10) {
                iterasi++;
                //kali = 0;
                TimestampAwal = TimestampAwal + 10;

            }*/
            //TV3.setText(String.valueOf(kali));
            //TV3.setText(String.valueOf(adaPass));
            kali++;
            //fallDetected= false;
            //writeNewData(valueUsername, iterasi, kali,alphaMagnitude,
omegaMagnitude, thetaMagnitude, fallDetected,  Timestamp);
            //writeNewClothes(alphaMagnitude, omegaMagnitude,
thetaMagnitude, minalphaMagnitude, waktu, kali);

            sample++;

            //if(Timestamp>=TimestampAwal+10){
            kaliTotal = kali;
            //kaliTV.setText(String.valueOf(kaliTotal));
            //onPause();
            //for(int i=0; i<kaliTotal;i++){

            //}
            /*
                for(int i =0; i<kaliTotal; i++){
                    Alpha = con[i][0];
                    Omega = con[i][1];
                    Theta = con[i][2];
                    Minalpha = con[i][3];
                    Waktu = con[i][4];
                    writeNewPost(Alpha, Omega, Theta, Minalpha, Waktu);
                }*/
```

```
            // }
            //waktu.add(alpha_THRESHOLD);
            //if(Timestamp==TimestampAwal+2)


        } catch (Exception e) {
            Toast.makeText(coba2.this, e.getMessage() + " oioi",
Toast.LENGTH_SHORT).show();
            Log.d(TAG, "onSensorChanged: " + e.toString());
        }
        //medianFilterAcc.reset();
        //medianFilterGyro.reset();
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int i) {

    }

    @Override
    protected void onResume() {
        super.onResume();
        if(!isOFF){
            mSensor.registerListener(coba2.this, sensorA,
SensorManager.SENSOR_DELAY_NORMAL);
            mSensor.registerListener(coba2.this, sensorB,
SensorManager.SENSOR_DELAY_NORMAL);
        }


        //mSensor.registerListener(coba2.this, sensorA, 500000);
        //mSensor.registerListener(coba2.this, sensorB, 500000);

    }

    /*
    @Override
    protected void onPause() {
        super.onPause();
        mSensor.unregisterListener(coba2.this);
    }*/


}
```

Lampiran 7 Pembentukan Model

```
x = Y;
t = targett;

trainFcn = 'traingdx';
hiddenLayerSize = 50;
net = feedforwardnet(hiddenLayerSize, trainFcn);
    ut.processFcns = {'removeconstantrows', 'mapstd',
    nax'};
    ideFcn = 'dividerand';
    ideMode = 'sample';
    ideParam.trainRatio = 90/100;
    ideParam.valRatio = 10/100;
```

```
net.divideParam.testRatio = 0/100;
net.trainParam.epochs=5000;
net.trainParam.goal=0;
net.trainParam.max_fail = 8;
net.trainParam.lr=0.001;
net.trainParam.lr_inc=1.05;
net.trainParam.lr_dec=0.7;
net.trainParam.max_perf_inc = 1.04;
net.trainParam.mc=0.9;
net.trainParam.min_grad = 0.00001;
net.layers{1}.transferFcn = 'logsig';
net.layers{2}.transferFcn = 'purelin';
net.performFcn = 'mse';
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
    'plotconfusion', 'plotroc', 'plotregression', 'plotwb'};
[net,tr] = train(net,x,t);

Bobotakhir_input9 = net.IW{1,1};
Bobotakhir_bias_input9 = net.b{1,1};
Bobotakhir_layer9 = net.LW{2,1};
Bobotakhir_bias_layer9 = net.b{2,1};
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y)
tind = vec2ind(t);
yind = vec2ind(y);
percentErrors = sum(tind ~= yind)/numel(tind);
trainTargetsA = t .* tr.trainMask{1};
valTargetsA = t .* tr.valMask{1};
testTargetsW = t .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,y)
valPerformance = perform(net,valTargets,y)
testPerformance = perform(net,testTargets,y)
```