DAFTAR PUSTAKA

Analog Devices. (2012, July 25). *AD8232 Single-Lead, Heart Rate Monitor Front End*. Analog Devices, Inc.

Anisa, N. (2022, February 14). *Mengenal Artificial Neural Network*. https://sis.binus.ac.id/2022/02/14/mengenal-artificial-neural-network/

ASSH. (2016, December 25). *Body Anatomy: Upper Extremity Muscles | The Hand Society*. https://www.assh.org/handcare/safety/muscles

Brownlee, J. (2021, October 12). *Code Adam Optimization Algorithm From Scratch*. https://machinelearningmastery.com/adam-optimization-from-scratch/

Domino. (2023, October 24). *Feature Extraction*. https://domino.ai/data-science-dictionary/feature-extraction

Firmansyah, I., & Hayadi, B. H. (2022). Komparasi Fungsi Aktivasi Relu Dan Tanh Pada Multilayer Perceptron. *JIKO (Jurnal Informatika Dan Komputer)*, *6*(2), 200. https://doi.org/10.26798/jiko.v6i2.600

Heydarian, M., Doyle, T. E., & Samavi, R. (2022). MLCM: Multi-Label Confusion Matrix. *IEEE Access*, *10*, 19083–19095. https://doi.org/10.1109/ACCESS.2022.3151048

Jayaweera, P. Y. (2021). *Design and Implementation of Electromyography (EMG) based Real-Time Pattern Recognition model for Prosthetic hand Control* [Liverpool John Moores University]. https://doi.org/10.31219/osf.io/rd2cf

Kingma, D. P., & Ba, J. (2014). *Adam: A Method for Stochastic Optimization*.

Lim, V., Haryanto, W. E., & Salvirah. (2022, November 3). *Elektromiografi (EMG) - Fungsi, Prosedur dan Efek Samping*. Siloam Hospitals.

Mokhlesabadifarahani, B., & Gunjan, V. K. (2015). *EMG Signals Characterization in Three States of Contraction by Fuzzy Network and Feature Extraction*. Springer Singapore. https://doi.org/10.1007/978-981-287-320-0

Ruder, S. (2016). *An overview of gradient descent optimization algorithms*.

Salamat, L. (2017). *Desain Tangan Bionik Sebagai Protesa Tangan Berbasis Sinyal Otot* [Skripsi]. Universitas Airlangga.

Shaw, L., & Bhaga, S. (2012). Online EMG Signal Analysis for diagnosis of Neuromuscular diseases by using PCA and PNN. *International Journal Of Engineering Science and Technology 0975-5462*, *4*, 4453–4459.

LAMPIRAN

1. Program Pembacaan Data Sensor dan Pengiriman Data Bluetooth pada Mikrokontroler

```
#include "BluetoothSerial.h"

String device_name = "EMG Device";

#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif

#if !defined(CONFIG_BT_SPP_ENABLED)
#error Serial Bluetooth not available or not enabled. It is only available for the ESP32
chip.
#endif

// AD8232 PIN
#define OUT 27
#define LOm 26
#define LOp 25

BluetoothSerial SerialBT;

// Handle received and sent message
String message = "";
String serialStr = "";
char incomingChar;

int value = 0;

String valueString = "";

bool reading = false;
bool showing = false;
```

```
);
ce_name); //Bluetooth device name

/ Byte array to hold the MAC address from getBtAddress()
```

```
  BTAddress mac_obj; // Object holding instance of BTAddress with the MAC (for more details see libraries/BluetoothSerial/src/BTAddress.h)
  String mac_str; // String holding the text version of MAC in format AA:BB:CC:DD:EE:FF

  SerialBT.getBtAddress(mac_arr); // Fill in the array
  mac_obj = SerialBT.getBtAddressObject(); // Instantiate the object
  mac_str = SerialBT.getBtAddressString(); // Copy the string

  Serial.print("This device is instantiated with name "); Serial.println(device_name);
  Serial.print("The mac address using string: "); Serial.println(mac_str.c_str());

  // Init AD8232
  pinMode(LOm, INPUT);
  pinMode(LOp, INPUT);
}

void loop() {
  // Get data sensor
  if (!((digitalRead(LOm) == 1) || (digitalRead(LOp) == 1))) {
    value = analogRead(OUT);
  }

  // Start send data sensor if reading is true
  if (reading) {
    valueString = String(value);
    Serial.println(valueString);
    SerialBT.println(valueString);
  }

  // Read received messages
  if (SerialBT.available()) {
    char incomingChar = SerialBT.read();
    if (incomingChar != '\n'){
      message += String(incomingChar);
    }
    else{
                                    age);


                                    is "#Start#"
                                    art#") {
```

```
    reading = true;
  }

  // Check if message is "#Stop#"
  if (message == "#Stop#") {
    reading = false;
  }

  // Read serial message
  if (Serial.available()) {
    serialStr = Serial.readString();
    int index = serialStr.length();
    serialStr.remove(index-1);
  }

  // Check if serial message is "#Start#"
  if (serialStr == "#Start#") {
    showing = true;
  }

  // Check if serial message is "#Stop#"
  if (serialStr == "#Stop#") {
    showing = false;
  }

  // Start show data sensor if showing is true
  if (showing) {
    valueString = String(value);
    Serial.println(valueString);
  }

  delay(1);
}
```

2. Program Pembacaan Data Bluetooth pada Aplikasi VR

```
public void ReadDataSensor()
{
    if (Application.platform != RuntimePlatform.Android)
        return;

    datas = new List<string>();
    dataSize = 15000;
    dataIter = 0;

    if (isConnected)
    {
        display.text += "Wait for reading...\n";

        isReading = true;
        BluetoothConnector.CallStatic("WriteData", "#Start#\n");
    }
}

public void ReadData(string data)
{
    Debug.Log("BT Stream: " + data);

    if (isReading)
    {
        dataReceived.text = data;
        datas.Add(data);
        dataIter++;
        float recordProgressVal = (float) dataIter / (float) dataSize;
        recordProgress.text = recordProgressVal.ToString();

        // stop if data size completed
        if (dataIter == dataSize)
        {
            isReading = false;
            BluetoothConnector.CallStatic("WriteData", "#Stop#\n");
                                = "Sensor data reading completed\n";
```

3. Program Ekstraksi Fitur

Mean (ME)

```
public double ME(List<double> datas, int? num = null)
{
    double data = 0;
    foreach (var dat in datas)
    {
        data += dat;
    }
    data /= datas.Count;
    if (num != null)
        data = Simplification(data, (int)num);

    return data;
}
public List<double> ME(List<double> datas, int length, int? num = null)
{
    var newDatas = new List<double>();
    for (int i = 0; i < datas.Count; i += length)
    {
        var dataRange = datas.GetRange(i, length);
        var me = ME(dataRange, num);
        newDatas.Add(me);
    }

    return newDatas;
}
```

Root Mean Square (RMS)

```
public double RMS(List<double> datas, int? num = null)
{
    double data = 0;
    foreach (var dat in datas)
    {
```

```
w(dat, 2);

ata);
t;

tion(data, (int)num);
```

```
    return data;
}
public List<double> RMS(List<double> datas, int length, int? num = null)
{
    var newDatas = new List<double>();
    for (int i = 0; i < datas.Count; i += length)
    {
        var dataRange = datas.GetRange(i, length);
        var rms = RMS(dataRange, num);
        newDatas.Add(rms);
    }

    return newDatas;
}
```

Slope Sign Change (SSC)

```
public double SSC(List<double> datas, int? num = null)
{
    double data = 0;
    for (int j = 1; j < datas.Count - 2; j++)
    {
        data += FunSSC((datas[j+1] - datas[j]) * (datas[j] - datas[j-1]));
    }
    if (num != null)
        data = Simplification(data, (int)num);

    return data;
}
public List<double> SSC(List<double> datas, int length, int? num = null)
{
    var newDatas = new List<double>();
    for (int i = 0; i < datas.Count; i += length)
    {
        var dataRange = datas.GetRange(i, length);
        var ssc = SSC(dataRange, num);
                                    sc);
```

Simple Square Integral (SSI)

```
public double SSI(List<double> datas, int? num = null)
{
    double data = 0;
    foreach (var dat in datas)
    {
        data += Math.Pow(dat, 2);
    }
    data /= datas.Count;
    if (num != null)
        data = Simplification(data, (int)num);

    return data;
}
public List<double> SSI(List<double> datas, int length, int? num = null)
{
    var newDatas = new List<double>();
    for (int i = 0; i < datas.Count; i += length)
    {
        var dataRange = datas.GetRange(i, length);
        var ssi = SSI(dataRange, num);
        newDatas.Add(ssi);
    }

    return newDatas;
}
```

Variance (VAR)

```
public double VAR(List<double> datas, int? num = null)
{
    double data = 0;
    foreach (var dat in datas)
    {
        data += Math.Pow(dat, 2);
    }
```

```
ıt - 1;

ıtion(data, (int)num);
```



```
AR(List<double> datas, int length, int? num = null)
```

```
{
    var newDatas = new List<double>();
    for (int i = 0; i < datas.Count; i += length)
    {
        var dataRange = datas.GetRange(i, length);
        var data = VAR(dataRange, num);
        newDatas.Add(data);
    }

    return newDatas;
}
```

Waveform Length (WL)

```
public double WL(List<double> datas, int? num = null)
{
    double data = 0;
    for (int j = 0; j < datas.Count - 1; j++)
    {
        data += Math.Abs(datas[j+1] - datas[j]);
    }
    if (num != null)
        data = Simplification(data, (int)num);

    return data;
}
public List<double> WL(List<double> datas, int length, int? num = null)
{
    var newDatas = new List<double>();
    for (int i = 0; i < datas.Count; i += length)
    {
        var dataRange = datas.GetRange(i, length);
        var wl = WL(dataRange, num);
        newDatas.Add(wl);
    }

    return newDatas;
```

4. Program Pemrosesan Data dengan Ekstraksi Fitur

```
int meNum = 3;
int rmsNum = 2;
int sscNum = 0;
int ssiNum = 6;
int varNum = 6;
int wlNum = 4;

int feLength = 150;
```

Ekstraksi fitur berurut

```
// feature extraction: Mean
var meFingerIndex = FeatureExtraction.ME(doubleFingerIndex, feLength, num:
    meNum);
var meFingerThumb = FeatureExtraction.ME(doubleFingerThumb, feLength, num:
    meNum);
var meHandClose = FeatureExtraction.ME(doubleHandClose, feLength, num:
    meNum);
var meHandOpen = FeatureExtraction.ME(doubleHandOpen, feLength, num:
    meNum);

// feature extraction: Root Mean Square
var rmsFingerIndex = FeatureExtraction.RMS(doubleFingerIndex, feLength, num:
    rmsNum);
var rmsFingerThumb = FeatureExtraction.RMS(doubleFingerThumb, feLength, num:
    rmsNum);
var rmsHandClose = FeatureExtraction.RMS(doubleHandClose, feLength, num:
    rmsNum);
var rmsHandOpen = FeatureExtraction.RMS(doubleHandOpen, feLength, num:
    rmsNum);

// feature extraction: Slope Sign Change
var sscFingerIndex = FeatureExtraction.SSC(doubleFingerIndex, feLength, num:
    sscNum);
var sscFingerThumb = FeatureExtraction.SSC(doubleFingerThumb, feLength, num:
                                           = FeatureExtraction.SSC(doubleHandClose, feLength, num:
                                           = FeatureExtraction.SSC(doubleHandOpen, feLength, num:
```

```
// feature extraction: Simple Square Integral
var ssiFingerIndex = FeatureExtraction.SSI(doubleFingerIndex, feLength, num:
    ssiNum);
var ssiFingerThumb = FeatureExtraction.SSI(doubleFingerThumb, feLength, num:
    ssiNum);
var ssiHandClose = FeatureExtraction.SSI(doubleHandClose, feLength, num:
    ssiNum);
var ssiHandOpen = FeatureExtraction.SSI(doubleHandOpen, feLength, num:
    ssiNum);

// feature extraction: Variace
var varFingerIndex = FeatureExtraction.VAR(doubleFingerIndex, feLength, num:
    varNum);
var varFingerThumb = FeatureExtraction.VAR(doubleFingerThumb, feLength, num:
    varNum);
var varHandClose = FeatureExtraction.VAR(doubleHandClose, feLength, num:
    varNum);
var varHandOpen = FeatureExtraction.VAR(doubleHandOpen, feLength, num:
    varNum);

// feature extraction: Waveform Length
var wlFingerIndex = FeatureExtraction.WL(doubleFingerIndex, feLength, num:
    wlNum);
var wlFingerThumb = FeatureExtraction.WL(doubleFingerThumb, feLength, num:
    wlNum);
var wlHandClose = FeatureExtraction.WL(doubleHandClose, feLength, num:
    wlNum);
var wlHandOpen = FeatureExtraction.WL(doubleHandOpen, feLength, num:
    wlNum);
```

Ekstraksi Fitur Acak

```
int dataCount = 15000;
int extrCount = (dataCount / feLength) * 5;

// run feature extraction
int iter = 0;
```

```
t)

ı.Range(0, dataCount - feLength);
```



```
x = doubleFingerIndex.GetRange(index, feLength);
```

```
var dataFingerThumb = doubleFingerThumb.GetRange(index, feLength);
var dataHandClose = doubleHandClose.GetRange(index, feLength);
var dataHandOpen = doubleHandOpen.GetRange(index, feLength);

// fe: Mean
var meFingerIndex = FeatureExtraction.ME(dataFingerIndex, num: meNum);
var meFingerThumb = FeatureExtraction.ME(dataFingerThumb, num: meNum);
var meHandClose = FeatureExtraction.ME(dataHandClose, num: meNum);
var meHandOpen = FeatureExtraction.ME(dataHandOpen, num: meNum);

// fe: Root Mean Square
var rmsFingerIndex = FeatureExtraction.RMS(dataFingerIndex, num: rmsNum);
var rmsFingerThumb = FeatureExtraction.RMS(dataFingerThumb, num:
   rmsNum);
var rmsHandClose = FeatureExtraction.RMS(dataHandClose, num: rmsNum);
var rmsHandOpen = FeatureExtraction.RMS(dataHandOpen, num: rmsNum);

// fe: Slope Sign Change
var sscFingerIndex = FeatureExtraction.SSC(dataFingerIndex, num: sscNum);
var sscFingerThumb = FeatureExtraction.SSC(dataFingerThumb, num: sscNum);
var sscHandClose = FeatureExtraction.SSC(dataHandClose, num: sscNum);
var sscHandOpen = FeatureExtraction.SSC(dataHandOpen, num: sscNum);

// fe: Simple Square Integral
var ssiFingerIndex = FeatureExtraction.SSI(dataFingerIndex, num: ssiNum);
var ssiFingerThumb = FeatureExtraction.SSI(dataFingerThumb, num: ssiNum);
var ssiHandClose = FeatureExtraction.SSI(dataHandClose, num: ssiNum);
var ssiHandOpen = FeatureExtraction.SSI(dataHandOpen, num: ssiNum);

// fe: Variance
var varFingerIndex = FeatureExtraction.VAR(dataFingerIndex, num: varNum);
var varFingerThumb = FeatureExtraction.VAR(dataFingerThumb, num: varNum);
var varHandClose = FeatureExtraction.VAR(dataHandClose, num: varNum);
var varHandOpen = FeatureExtraction.VAR(dataHandOpen, num: varNum);

// fe: Wave Length
var wlFingerIndex = FeatureExtraction.WL(dataFingerIndex, num: wlNum);
) = FeatureExtraction.WL(dataFingerThumb, num: wlNum);
: FeatureExtraction.WL(dataHandClose, num: wlNum);
 FeatureExtraction.WL(dataHandOpen, num: wlNum);
```

5. Program Pengacakan

Fisher Yates

```
public List<string> FisherYates(List<string> _datas)
{
    var n = _datas.Count;
    var rand = new System.Random();

    for (int i = n - 1; i > 0; i--)
    {
        var k = rand.Next(i + 1);
        (_datas[i], _datas[k]) = (_datas[k], _datas[i]);
    }

    return _datas;
}
```

6. Program Pengkodean Label

One Hot Encoder

```
private double[,] OneHotEncoder(string[] _labels)
{
    string[] sortLabels = new string[_labels.Length];
    for (int i = 0; i < _labels.Length; i++)
    {
        sortLabels[i] = _labels[i];
    }
    Array.Sort(sortLabels);

    string label = "";
    singleLabels = new List<string>();
    foreach (var item in sortLabels)
    {
        if (label != item)
```

```
Add(item);
```



```
w Dictionary<string, double[]>();
```

```
    for (int i = 0; i < singleLabels.Count; i++)
    {
        var oneHot = new double[singleLabels.Count];
        for (int j = 0; j < oneHot.Length; j++)
        {
            if (j == i)
            {
                oneHot[j] = 1;
            }
            else
            {
                oneHot[j] = 0;
            }
        }
        dictLabels.Add(singleLabels[i], oneHot);
    }

    var newLabels = new double[_labels.Length, singleLabels.Count];
    for (int i = 0; i < _labels.Length; i++)
    {
        var oneHotLabel = dictLabels[_labels[i]];
        for (int j = 0; j < oneHotLabel.Length; j++)
        {
            newLabels[i,j] = oneHotLabel[j];
        }
    }

    return newLabels;
}
```

7. Program Fungsi Aktivasi

Rectified Linear Unit (ReLu)

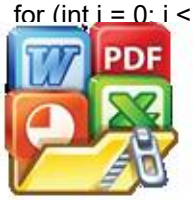
```
private double ReLu(double x)
{
    return System.Math.Max(0, x);
}
```

Softmax

```
private double[] Softmax(double[] array)
{
    var result = new double[array.Length];
    var arrayExp = new double[array.Length];

    for (int i = 0; i < array.Length; i++)
    {
        arrayExp[i] = Math.Exp(array[i]);
    }

    for (int i = 0; i < array.Length; i++)
    {
        result[i] = arrayExp[i] / arrayExp.Sum();
    }

    return result;
}
```

Argmax

```
private double[] Argmax(double[] array)
{
    var result = new double[array.Length];
    double maxNum = array.Max();

    for (int i = 0; i < array.Length; i++)
```



```
                                        xNum)
```

```
        {
            result[i] = 0;
        }
    }

    return result;
}
```

Turunan (*Derivative*) ReLu

```
private double ReLuDerivative(double x)
{
    if (x < 0)
    {
        return 0;
    }
    else
    {
        return 1;
    }
}
```

8. Program Fungsi Cost/Loss

Categorical Cross Entropy

```
private double[] CategoricalCrossEntropy(double[,] actualVal, double[,] predictVal)
{
    var result = new double[predictVal.GetLength(0)];
    for (int i = 0; i < predictVal.GetLength(0); i++)
    {
        var cce = new double[predictVal.GetLength(1)];
        for (int j = 0; j < predictVal.GetLength(1); j++)
        {
            cce[j] = actualVal[i,j] * Math.Log(predictVal[i,j]);
        }
```

um();

9. Program Fungsi Optimasi Neural Network

ADAM (Adaptive Movement Estimation) Optimization

```
private void ADAMWH(double[,] w, double[,] w_der, int t)
{
   var dw = Transpose(w_der);

   for (int i = 0; i < w.GetLength(0); i++)
   {
      for (int j = 0; j < w.GetLength(1); j++)
      {
         var gt = dw[i,j];
         mwh1[i,j] = beta1 * mwh1[i,j] + (1 - beta1) * gt;
         mwh2[i,j] = beta2 * mwh2[i,j] + (1 - beta2) * Math.Pow(gt, 2);

         var m1_hat = mwh1[i,j] / (1 - Math.Pow(beta1, t));
         var m2_hat = mwh2[i,j] / (1 - Math.Pow(beta2, t));

         w[i,j] = w[i,j] - (alpha * m1_hat / (Math.Sqrt(m2_hat) + epsilon));
      }
   }
}

private void ADAMWO(double[,] w, double[,] w_der, int t)
{
   var dw = Transpose(w_der);

   for (int i = 0; i < w.GetLength(0); i++)
   {
      for (int j = 0; j < w.GetLength(1); j++)
      {
         var gt = dw[i,j];
         mwo1[i,j] = beta1 * mwo1[i,j] + (1 - beta1) * gt;
         mwo2[i,j] = beta2 * mwo2[i,j] + (1 - beta2) * Math.Pow(gt, 2);

         var m1_hat = mwo1[i,j] / (1 - Math.Pow(beta1, t));
                      mwo2[i,j] / (1 - Math.Pow(beta2, t));

                     (alpha * m1_hat / (Math.Sqrt(m2_hat) + epsilon));
```

```
private void ADAMBH(double[] b, double[] b_der, int t)
{
    var db = b_der;

    for (int i = 0; i < b.Length; i++)
    {
        var gt = db[i];
        mbh1[i] = beta1 * mbh1[i] + (1 - beta1) * gt;
        mbh2[i] = beta2 * mbh2[i] + (1 - beta2) * System.Math.Pow(gt, 2);

        var m1_hat = mbh1[i] / (1 - Math.Pow(beta1, t));
        var m2_hat = mbh2[i] / (1 - Math.Pow(beta2, t));

        b[i] = b[i] - (alpha * m1_hat / (System.Math.Sqrt(m2_hat) + epsilon));
    }
}

private void ADAMBO(double[] b, double[] b_der, int t)
{
    var db = b_der;

    for (int i = 0; i < b.Length; i++)
    {
        var gt = db[i];
        mbo1[i] = beta1 * mbo1[i] + (1 - beta1) * gt;
        mbo2[i] = beta2 * mbo2[i] + (1 - beta2) * System.Math.Pow(gt, 2);

        var m1_hat = mbo1[i] / (1 - Math.Pow(beta1, t));
        var m2_hat = mbo2[i] / (1 - Math.Pow(beta2, t));

        b[i] = b[i] - (alpha * m1_hat / (System.Math.Sqrt(m2_hat) + epsilon));
    }
}
```

10. Program Pelatihan Model Klasifikasi JST

```
int featureCount = 6;
int labelCount = 4;
int nodeIn = featureCount;
int nodeH = 12;
int nodeOut = labelCount;

double alpha = 0.001;
double beta1 = 0.9;
double beta2 = 0.999;
double epsilon = 1E-8;
```

Inisiasi parameter

```
private void InitWeight(double[,] weight)
{
    int row = weight.GetLength(0);
    int col = weight.GetLength(1);

    var rand = new Random(0);

    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            weight[i,j] = rand.NextDouble();
        }
    }
}

private void InitBias(double[] bias)
{
    int length = bias.Length;

    var rand = new Random(0);
```

```
gth; i++)

xtDouble();
```

Pelatihan (*Train*)

```
int _epoch_lim = 10000;
int _loss_lim = 0.001;

public async Task Train(double[,] _inputTrain, string[] _labelTrain, int _epoch_lim,
double _loss_lim)
{
    // get time
    var last = DateTime.Now;

    // init data train
    inputTrain = _inputTrain;
    labelTrain = OneHotEncoder(_labelTrain);

    int _epoch = 1;
    double _loss = 1;

    await Task.Run(() =>
    {
        // train while epoch < epoch_lim and loss > loss_lim
        while (_epoch < _epoch_lim + 1 && _loss > _loss_lim)
        {

            /* FEED FORWARD */

            // hidden layer
            var zH = AddArray(Dot(inputTrain, wH), bH);
            var aH = new double[zH.GetLength(0), zH.GetLength(1)];
            for (int i = 0; i < zH.GetLength(0); i++)
            {
                for (int j = 0; j < zH.GetLength(1); j++)
                {
                    aH[i,j] = ReLu(zH[i,j]);
                }
            }
```



```
dArray(Dot(aH, wOut), bOut);
w double[zOut.GetLength(0), zOut.GetLength(1)];
< zOut.GetLength(0); i++)

rr = new double[zOut.GetLength(1)];
; j < zOut.GetLength(1); j++)
```

```
        {
            singleArr[j] = zOut[i,j];
        }

        var sofmaxVal = Softmax(singleArr);
        for (int j = 0; j < zOut.GetLength(1); j++)
        {
            aOut[i,j] = sofmaxVal[j];
        }
    }

    // loss
    lossArr = CategoricalCrossEntropy(labelTrain, aOut);
    _loss = lossArr.Sum() / lossArr.Length;
    loss = _loss;

    // print loss
    if (_epoch % 100 == 0)
        UnityEngine.Debug.Log($"Epoch: {_epoch} | Loss: {_loss}");


    /* BACK PROPAGATION */

    // output layer
    var dcost_dzout = Transpose(Sub(aOut, labelTrain));
    var dzout_dwout = aH;
    var dcost_dwout = Dot(dcost_dzout, dzout_dwout);
    var dcost_dbout = new double[dcost_dzout.GetLength(0)];
    for (int i = 0; i < dcost_dzout.GetLength(0); i++)
    {
        var singleArr = new double[dcost_dzout.GetLength(1)];
        for (int j = 0; j < dcost_dzout.GetLength(1); j++)
        {
            singleArr[j] = dcost_dzout[i,j];
        }
        dcost_dbout[i] = singleArr.Sum();
    }
```

```
                    = wOut;
                    = new double[zH.GetLength(0), zH.GetLength(1)];
                    < zH.GetLength(0); i++)

                    ; j < zH.GetLength(1); j++)
```

```
            {
                dah_dzh[i,j] = ReLuDerivative(zH[i,j]);
            }
        }
        var dzh_dwh = inputTrain;
        var dcost_dah = Dot(dzout_dah, dcost_dzout);
        var dcost_dzh = Multiply(dcost_dah, Transpose(dah_dzh));
        var dcost_dwh = Dot(dcost_dzh, dzh_dwh);
        var dcost_dbh = new double[dcost_dzh.GetLength(0)];
        for (int i = 0; i < dcost_dzh.GetLength(0); i++)
        {
            var singleArr = new double[dcost_dzh.GetLength(1)];
            for (int j = 0; j < dcost_dzh.GetLength(1); j++)
            {
                singleArr[j] = dcost_dzh[i,j];
            }
            dcost_dbh[i] = singleArr.Sum();
        }


        /* OPTIMIZATION */

        // hidden layer
        ADAMWH(wH, dcost_dwh, _epoch);
        ADAMBH(bH, dcost_dbh, _epoch);

        // output layer
        ADAMWO(wOut, dcost_dwout, _epoch);
        ADAMBO(bOut, dcost_dbout, _epoch);

        // update epoch
        epoch = _epoch;
        _epoch++;
    }

    // get time
    var current = DateTime.Now;
                last).TotalSeconds;
```

Validasi (*Validate*)

```
public async Task Validate(double[,] _inputTest, string[] _labelTest)
{
    // init data test
    inputTest = _inputTest;
    labelTest = OneHotEncoder(_labelTest);
    labelTestLing = _labelTest;

    predictedLabels = new string[labelTestLing.Length];
    int predictedTrue = 0;

    var labCount = singleLabels.Count;
    var lab = new int[labCount];
    var tp = new double[labCount];
    var tn = new double[labCount];
    var fp = new double[labCount];
    var fn = new double[labCount];

    await Task.Run(() =>
    {
        for (int i = 0; i < inputTest.GetLength(0); i++)
        {
            var input = new double[inputTest.GetLength(1)];
            for (int j = 0; j < inputTest.GetLength(1); j++)
            {
                input[j] = inputTest[i,j];
            }

            predictedLabels[i] = Predict(input);
            var actLabel = labelTestLing[i];
            var predLabel = predictedLabels[i];

            for (int j = 0; j < labCount; j++)
            {
                if (actLabel == singleLabels[j]) lab[j]++;
            }
                                                    = predLabel)

                                                    ue++;
                                                    ; j < labCount; j++)

                                                    el == singleLabels[j]) tp[j]++;
```

```
                else tn[j]++;
            }
        }
        else
        {
            for (int j = 0; j < labCount; j++)
            {
                if (actLabel == singleLabels[j]) fn[j]++;
                if (predLabel == singleLabels[j]) fp[j]++;
            }
        }
    }
});

// accuracy
accuracy = 100.0 * predictedTrue / predictedLabels.Length;

// calc
var precision_i = new double[labCount];
var recall_i = new double[labCount];
for (int i = 0; i < labCount; i++)
{
    precision_i[i] = tp[i] / (tp[i] + fp[i]);
    recall_i[i] = tp[i] / (tp[i] + fn[i]);
}

// labels
int labSum = 0;
foreach (var lab_i in lab)
{
    labSum += lab_i;
}

// precision
for (int i = 0; i < precision_i.Length; i++)
{
    precision_macro += precision_i[i];
                ted += lab[i] * precision_i[i];

                labCount;
                /= labSum;

                all_i.Length; i++)
```

```
    {
        recall_macro += recall_i[i];
        recall_weighted += lab[i] * recall_i[i];
    }
    recall_macro /= labCount;
    recall_weighted /= labSum;
}
```

Prediksi (*Predict*)

```
public string Predict(double[] input)
{
    var newInput = ArrayToMatrix("row", input);

    var zH = AddArray(Dot(newInput, wH), bH);
    var aH = new double[zH.GetLength(0), zH.GetLength(1)];
    for (int i = 0; i < zH.GetLength(0); i++)
    {
        for (int j = 0; j < zH.GetLength(1); j++)
        {
            aH[i,j] = ReLu(zH[i,j]);
        }
    }

    var zOut = AddArray(Dot(aH, wOut), bOut);
    var new_zOut = MatrixToArray(zOut);

    var sofmaxVal = Softmax(new_zOut);
    var argmaxVal = Argmax(sofmaxVal);
    var aOut = argmaxVal.ToList();

    var index = 0;
    for (int i = 0; i < argmaxVal.Length; i++)
    {
        if (argmaxVal[i] == 1)
        {
            index = i;
```

```
                = singleLabels[index];

            bel;
```

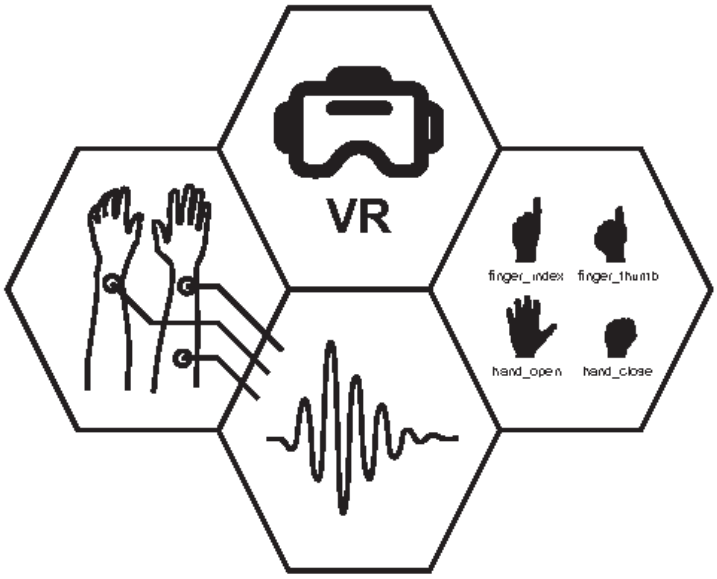
11. Program Prediksi Data Secara Realtime

```
void Update()
{
    modelLabel.text = modelLabelVal;
    if (EMGData.IsStreaming)
    {
        try
        {
            dataReceived.text = EMGData.DataStream.Count.ToString();
            modelLabelVal = PredictModel(EMGData.DataStream);
        }
        catch (System.Exception ex)
        {
            display.text += ex.Message + "\n";
        }
    }
}
```

12. Buku Pedoman (Manual Book)

BUKU PANDUAN

TEKOLOGI VIRTUALISASI PERGERAKAN JARI TANGAN BERBASIS SINYAL MIOELEKTRIK MENGGUNAKAN SENSOR ELEKTROMIOGRAFI DAN ALGORITMA JARINGAN SARAF TIRUAN



ERSITAS HASANUDDIN
ILTAS TEKNIK
JURUSAN TEKNIK INFORMATIKA

V1.0

Kata Pembuka

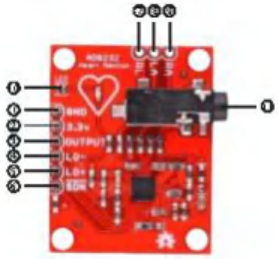
Pengguna yang Terhormat,

Ucapan terima kasih yang sebesar-besarnya kami sampaikan kepada Anda, para pengguna, atas kepercayaan Anda memilih dan menggunakan teknologi ini. Buku panduan ini dirancang untuk membantu Anda memahami dan memanfaatkan fitur-fitur alat dan aplikasi secara optimal. Kami berharap informasi yang disajikan di dalamnya dapat memudahkan Anda dalam mengoperasikan alat dengan aman dan efisien. Semoga panduan ini dapat menjadi referensi yang bermanfaat dalam setiap langkah penggunaan, sehingga Anda dapat meraih hasil terbaik dalam pekerjaan Anda. Terima kasih atas perhatian dan dukungan Anda, serta semoga teknologi ini dapat memenuhi harapan dan kebutuhan Anda.

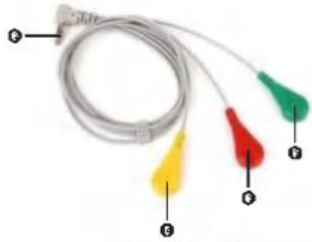
Daftar Isi

Pengenalan Perangkat

Modul AD8232



(Modul Sensor AD8232)



(Kabel Elektroda)

No	Nama	Kegunaan
1	GND	Pin ground
2	3.3v	Pin tegangan 3.3 Volt
3	OUTPUT	Pin keluaran sinyal
4	LO-	Pin Lead Off positif
5	LO+	Pin Lead Off negatif
6	SDN	Pin shutdown
7	RA	Pin Right Arm
8	LA	Pin Left Arm
9	RL	Pin Right Leg
10	LED	LED sinyal
11	Jack 3.5mm Female	Port jack kabel elektroda
12	Jack 3.5mm 3pin Male	Jack kabel elektroda
13	Elektroda RA	Elektroda Right Arm (Merah)
14	Elektroda LA	Elektroda Left Arm (Kuning)
15	Elektroda RL	Elektroda Right Leg (Hijau)

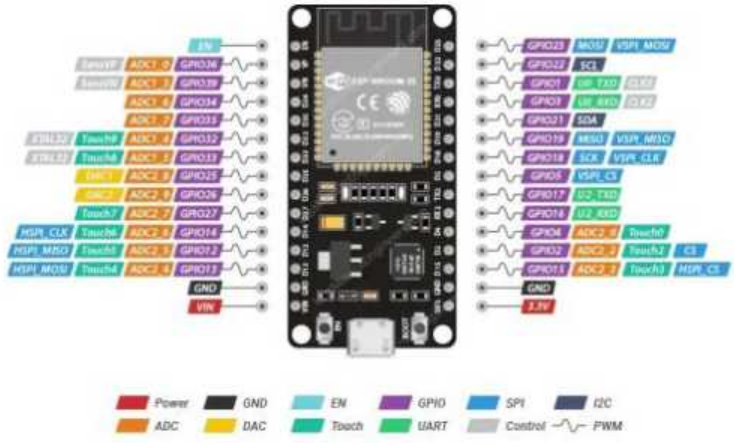
Elektroda Permukaan

(Tampak Atas)

(Tampak Bawah)

Pengenalan Perangkat

ESP32 DevKit V1



Sumber: LastMinuteEngineers.com

Oculus Quest

Manufacture	: Oculus	Device Type	: Standalone VR
Resolution	: 1440x1600 per-eye	Refresh Rate	: 72 Hz
WiFi	: WiFi 5	Bluetooth	: Bluetooth 5.0 LE
Operating System	: Android 10	Chipset	: Qualcomm Snapdragon 835
CPU	: Octa-core Kryo 280	GPU	: Adreno 540
Memory	: 4 GB	Storage	: 64 GB

Persiapan

Unggah Program Mikrokontroler

1. Unduh program mikrokontroler ESP32 pada link QR Code berikut:

2. Buka program yang telah diunduh dengan menggunakan software Arduino IDE.
   Software Arduino IDE dapat diunduh pada link: https://www.arduino.cc/en/software
3. Pastikan *board* ESP32 telah ter-*install* pada software Arduino IDE.
   Tata cara *install board* ESP32 pada Arduino IDE dapat dilihat pada link:
   https://docs.espressif.com/projects/arduino-esp32/en/latest/installing.html
4. Hubungkan *board* mikrokontroler ESP32 pada perangkat komputer Anda.
5. Sesuaikan *board* dan *port* mikrokontroler pada software Arduino IDE
6. *Verify* program untuk mengecek *error* pada program
7. Jika terdapat *error* pada program, silahkan hubungi pembuat
8. Jika tidak terdapat *error* pada program, *upload* program ke mikrokontroler
9. Mikrokontroler ESP32 siap digunakan untuk mengambil data sinyal mioelektrik

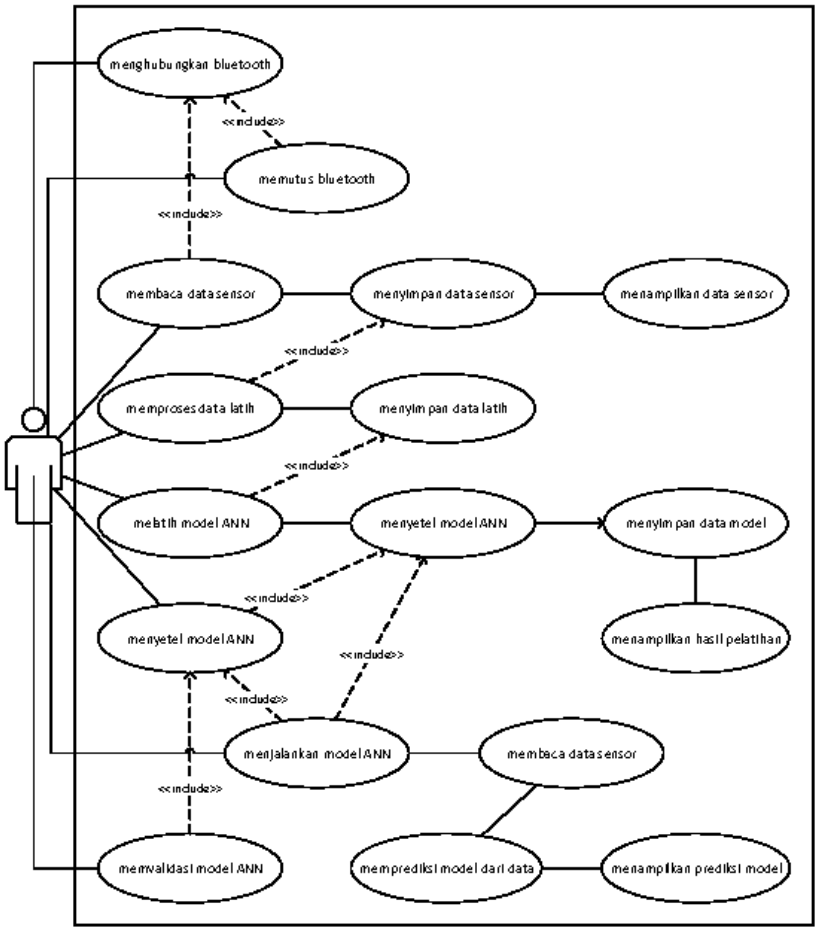
Pemasangan Aplikasi pada Perangkat VR

1. Unduh file aplikas VR Hand Classify pada link QR Code berikut:

2. Pasang aplikasi VR Hand Classify pada perangkat VR dengan aplikasi pihak ketiga, seperti:
   Meta Quest Developer Hub link: https://developer.oculus.com/documentation/unity/ts-odh/
   VRsideloader, link: https://headjack.io/knowledge-base/how-to-easily-sideload-a-vr-app-to-oculus-quest-2/
   SideQuest, link: https://partner.equalreality.com/learning-guides/vr-setup-guides/oculus-quest/installing-with-sidequest
   Atau dengan menggunakan metode pemasangan lainnya
3. Pastikan aplikasi Hand Classify terpasang dan dapat berjalan dengan baik pada perangkat VR Anda
4. Jika terdapat *error* pada aplikasi, silahkan hubungi pembuat
5. Jika tidak terdapat *error*, aplikasi VR Hand Classify siap untuk digunakan

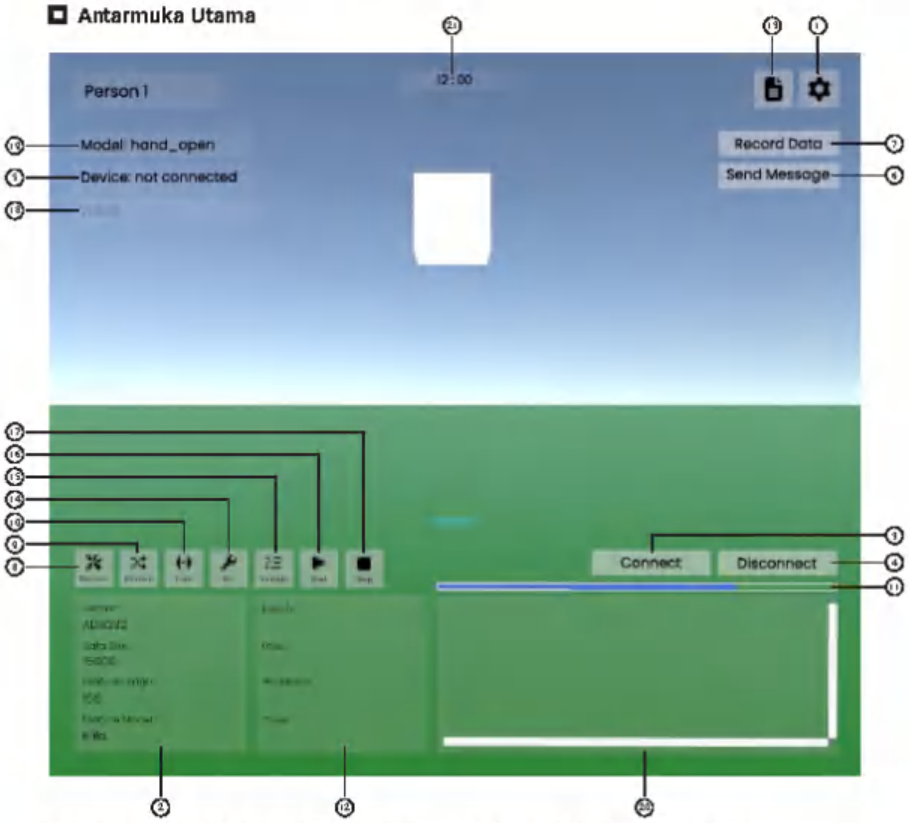
Catatan: Aplikasi Hand Classify merupakan aplikasi VR Android yang dikembangkan khusus untuk dijalankan pada perangkat Standalone VR Oculus Quest

Use Case Aplikasi VR



Antarmuka Aplikasi VR

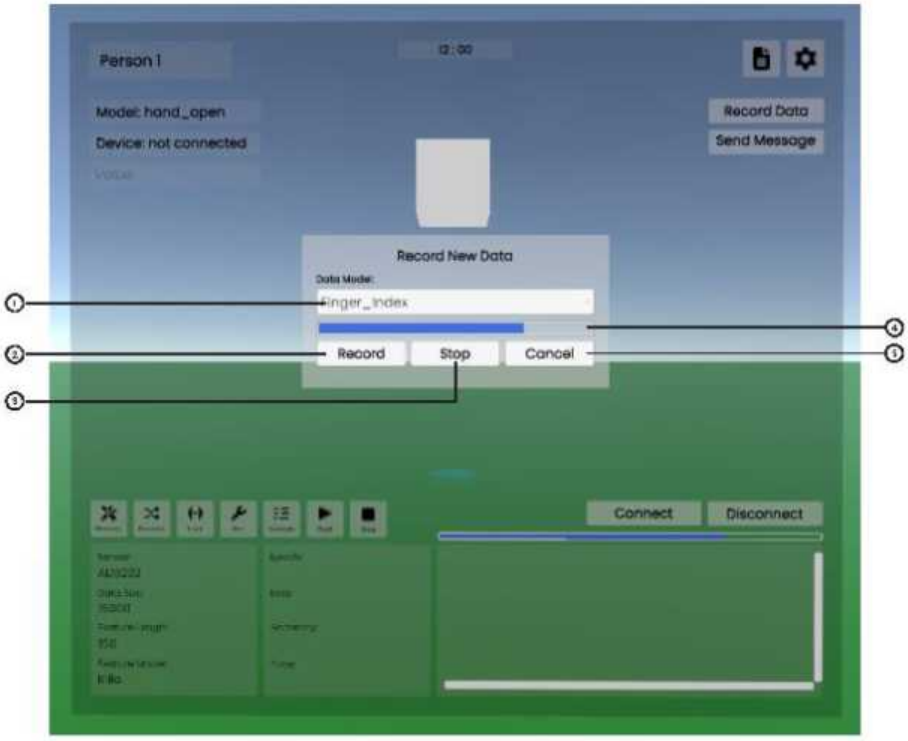
Antarmuka Utama



1. Tombol panel pengaturan aplikasi (pengaturan perangkat dan konfigurasi pelatihan)
2. Penampilan informasi konfigurasi pelatihan model
3. Tombol untuk menghubungkan koneksi bluetooth terhadap perangkat sensor
4. Tombol untuk memutuskan koneksi bluetooth terhadap perangkat sensor
5. Status koneksi bluetooth terhadap perangkat sensor
6. Tombol panel pengiriman pesan teks kepada perangkat sensor
7. Tombol panel pembacaan dan pengumpulan data sinyal sensor untuk setiap model gerak jari tangan
8. Tombol untuk memproses data mentah sinyal secara berurut menjadi data latih model
9. Tombol untuk memproses data mentah sinyal secara acak menjadi data latih model
10. Tombol untuk menjalankan proses pelatihan model klasifikasi
11. Bar progres pelatihan model klasifikasi
12. Penampilan informasi hasil pelatihan model klasifikasi
13. Tombol panel informasi histori pelatihan model
14. Tombol untuk menyetel model klasifikasi yang sedang aktif
15. Tombol untuk memperbaharui informasi model klasifikasi yang sedang aktif
16. Tombol untuk menjalankan proses prediksi kelas/label data sinyal sensor secara realtime
17. Tombol untuk menghentikan proses prediksi kelas/label data realtime
18. Data sinyal sensor yang diterima secara realtime
19. Hasil prediksi kelas/label data sinyal
20. Penampilan informasi aplikasi
21. Waktu aplikasi realtime sesuai zona waktu perangkat VR

Antarmuka Aplikasi VR

Panel Pengambilan/Pengumpulan Data

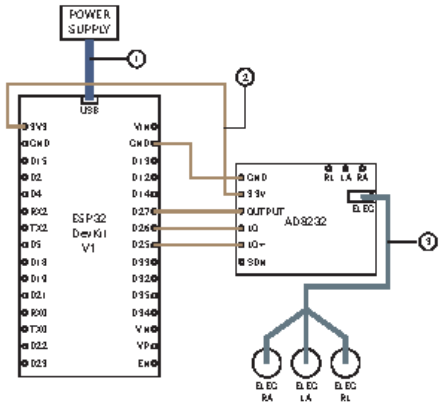


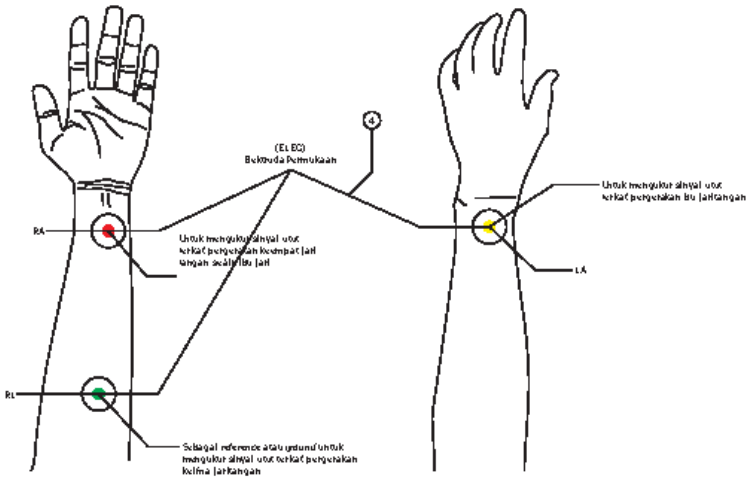
1. Pemilihan data model gerak jari tangan yang ingin diambil
2. Tombol untuk memulai pengambilan data sinyal sensor
3. Tombol untuk menghentikan pengambilan data
4. Bar progres pengumpulan data
5. Tombol untuk keluar dari panel pengambilan data

Langkah Penggunaan

Pemasangan Sensor

1. Nyalakan mikrkontroler ESP32 DevKit V1 dengan menghubungkan mikrokontroler ke sumber daya listrik
2. Hubungkan sensor AD8232 terhadap mikrokontroler ESP32 DevKit V1 dengan rangkaian seperti gambar di samping
3. Pasangkan elektroda permukaan pada sensor AD8232 dengan perantara kabel elektroda seperti pada gambar
4. Tempelkan ketiga elektroda pada lengan dengan posisi peletakan sebagaimana pada ilustrasi gambar di bawah
5. Gunakan lengan kiri untuk lebih memudahkan dalam pengambilan data
6. Rasakan pergerakan otot dengan menyentuh permukaan sebagaimana yang ditampilkan oleh gambar dengan sembari menggerakkan jari-jari tangan untuk memudahkan mendapatkan posisi elektroda yang tepat
7. Perhatikan LED sinyal pada sensor untuk memastikan sensor berjalan sesuai dengan pergerakan otot





BAGIAN BAWAH LENGAN

BAGIAN ATAS LENGAN

Langkah Penggunaan

Menghubungkan Koneksi Bluetooth pada Perangkat

1. Nyalakan mikrokontroler ESP32 DevKit V1
2. Pastikan program ESP32 telah terunggah (*upload*) pada mikrokontroler ESP32 DevKit V1
3. Pada perangkat VR, tambahkan perangkat bluetooth "EMG Device"
4. Pada aplikasi VR, sesuaikan nilai MAC Adress pada panel pengaturan dengan MAC Address bluetooth perangkat ESP32 DevKit V1
5. Hubungkan perangkat VR terhadap perangkat mikrokontroler ESP32 dengan menekan tombol Connect
6. Tunggu beberapa saat hingga status koneksi berubah menjadi "Connected" pada UI aplikasi
7. Jika status koneksi telah berubah menjadi "Connected", perangkat VR telah terhubung pada perangkat mikrokontroler ESP32 dan siap untuk melakukan pengambilan data
8. Jika status koneksi tidak berupa "Connected" atau status koneksi adalah "Disconnected", periksa apakah ada perangkat lain yang terhubung pada perangkat bluetooth "EMG Device" dan pastikan tidak terdapat perangkat lain yang sedang terkoneksi pada perangkat "EMG Device" kemudian koneksikan ulang perangkat VR terhadap perangkat mikrokontroler dengan menekan tombol "Connect"
9. Jika perangkat belum terhubung, pastikan MAC Address pada panel pengaturan telah sesuai dengan MAC Address bluetooth perangkat mikrokontroler ESP32 DevKit V1 yang digunakan
10. Jika perangkat masih belum terhubung, silahkan mulai ulang aplikasi VR dan *restart* perangkat mikrokontroler dengan menekan tombol "EN" pada *board* kemudian koneksikan ulang
11. Untuk menghentikan koneksi bluetooth perangkat VR terhadap perangkat mikrokontroler ESP32, tekan tombol "Disconnected" pada aplikasi VR

Pengambilan Data

1. Pastikan sensor AD8232 terhubung pada mikrokontroler ESP32 dan elektroda permukaan menempel pada lengan dengan baik dengan memperhatikan perubahaan terang/redup LED sinyal pada sensor saat menggerakkan jari-jari tangan
2. Untuk mengecek sinyal otot (mioelektrik) dari sensor diterima dengan baik, jalankan pembacaan data pada software Arduino IDE dengan memasukkan perintah "#Start#" pada serial monitor
3. Sambil pembacaan data berjalan, bentuk grafik data dapat dilihat dengan menggunakan fitur Serial Plotter pada software Arduino IDE
4. Jika pembacaan data sensor berjalan dengan baik, hentikan pengecekan dengan memasukkan perintah "#Stop#" pada serial monitor
5. Pada aplikasi VR, masuk pada panel pengumpulan data (panel Record) untuk mengumpulkan data sinyal otot (mioelektrik)
6. Pada panel, sesuaikan label data (Data Model) yang akan dikumpulkan
7. Tekan tombol "Record" untuk memulai pengumpulan data
8. Tunggu hingga proses pengambilan dan pengumpulan data selesai yang ditandai dengan penuhnya bar progres pengumpulan data dan informasi umpan balik yang ditampilkan pada panel penampil informasi aplikasi
9. Lakukan pengumpulan data berulang kali untuk setiap model hingga keempat model gerak jari tangan (finger_index, finger_thumb, hand_open, hand_close) terkumpul

Pelatihan Model Klasifikasi

1. Pastikan pengambilan data telah dilakukan dan data mentah sinyal otot (mioelektrik) dari keempat model gerak jari tangan (finger_index, finger_thumb, hand_open, hand_close) terlah dikumpulkan
2. Pada panel pengaturan (*settings*), atur konfigurasi pelatihan model sesuai yang diinginkan
3. Lakukan pemrosesan data mentah menjadi data latih dengan menekan tombol "Process" (untuk memproses data secara berurut) atau "Random" (untuk memproses data secara acak)
4. Jika pemrosesan data mentah menjadi data latih telah dilakukan (informasi timbal balik pada panel penampil informasi aplikasi), latih model klasifikasi dengan menekan tombol "Train"
5. Tunggu hingga proses pelatihan selesai (bar progres proses pelatihan model penuh atau infomasi timbalik balik pada panel informasi model dan pada panel penampil informasi aplikasi)
6. Untuk memastikan model telah berhasil dilatih, lakukan set dan validasi model dengan menekan tombol "Set" dan "Validate" secara berurut
7. Jika informasi akurasi model berhasil ditampilkan, maka model klasifikasi telah berhasil dilatih
8. Dengan selesainya pelatihan model klasifikasi, model dapat digunakan untuk memprediksi label data secara realtime dengan menekan tombol "Start" (untuk memulai klasifikasi data realtime) dan "Stop" (untuk menghentikan klasifikasi data realtime)

Kami berharap buku panduan ini telah memberikan pengetahuan yang komprehensif dan bermanfaat untuk Anda dalam menggunakan teknologi ini. Jika ada pertanyaan lebih lanjut atau kebutuhan akan informasi tambahan, jangan ragu untuk menghubungi kami. Terima kasih telah menggunakan teknologi kami, dan semoga teknologi ini dapat membantu meningkatkan produktivitas dan efisiensi Anda. Sukses selalu dalam setiap langkah pengembangan teknologi!!

TEKOLOGI VIRTUALISASI PERGERAKAN JARI TANGAN BERBASIS SINYAL MIOELEKTRIK MENGGUNAKAN SENSOR ELEKTROMIOGRAFI DAN ALGORITMA JARINGAN SARAF TIRUAN

Langkah baru menuju pengembangan teknologi dan pemanfaatan biopotensial yang lebih cerdas

PEMBUAT
email: ghazwul.s@gmail.com

UNIVERSITAS HASANUDDIN
I TEKNIK
HEN TEKNIK INFORMATIKA