IMPLEMENTASI SISTEM PENDETEKSI FIBRILASI ATRIUM BERDASARKAN INTERVAL DAN GRADIEN QRS MENGGUNAKAN METODE JARINGAN SARAF TIRUAN

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
Muhammad Bilal
NIM: 175150307111026

PROGRAM STUDI TEKNIK KOMPUTER
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2021

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

PENGESAHAN

IMPLEMENTASI SISTEM PENDETEKSI FIBRILASI ATRIUM BERDASARKAN INTERVAL DAN GRADIEN QRS MENGGUNAKAN METODE JARINGAN SARAF TIRUAN

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh :
Muhammad Bilal
NIM: 175150307111026

Skripsi ini telah diuji dan dinyatakan lulus pada
07 Mei 2021
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Rizal Maulana, S.T., M.T., M.Sc.	Eko Setiawan, S.T., M.T., Ph.D.
NIK: 201607 891009 1 001	NIK: 201102 870610 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Achmad Basuki, S.T., M.MG., Ph.D.
NIP: 19741118 200312 1 002

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

Repository Universitas Brawijaya

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar referensi.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 07 Mei 2021

METERAI TEMPEL
7BF4AJX172060136

Muhammad Bilal

NIM: 175150307111026

PRAKATA

Puji serta syukur penulis panjatkan kepada Allah SWT karena telah memberikan kelancaran dan kemudahan kepada penulis sehingga penulis dapat menyelesaikan penelitian dan penulisan laporan skripsi dengan judul "Implementasi Sistem Pendeteksi Fibrilasi Atrium Berdasarkan Interval dan Gradien QRS Menggunakan Metode Jaringan Saraf Tiruan". Hal tersebut dilakukan dengan tujuan untuk memenuhi persyaratan dalam menyelesaikan pendidikan sarjana program studi Teknik Komputer, Fakultas Ilmu Komputer, Universitas Brawijaya.

Dalam menyelesaikan penelitian dan penulisan laporan skripsi ini, tentunya terdapat cukup banyak masalah ataupun kendala yang dihadapi. Namun berkat dukungan, masukan maupun saran yang berasal dari berbagai pihak maka penulis berhasil menyelesaikan penelitian dan penulisan laporan skripsi ini dengan baik. Oleh karena itu, penulis ingin mengucapkan rasa terima kasih kepada berbagai pihak yang terlibat secara langsung maupun tidak langsung yang telah membantu menyelesaikan penelitian dan penulisan laporan skripsi ini. Pihak-pihak tersebut antara lain:

1. Kakak, Mama dan Papa selaku keluarga penulis yang telah memberikan do'a, semangat, motivasi serta sumber kekuatan kepada penulis untuk dapat menyelesaikan skripsi ini.
2. Bapak Rizal Maulana, S.T., M.T., M.Sc. selaku dosen pembimbing satu penulis yang bersedia meluangkan waktunya untuk memberikan arahan, masukan dan bimbingan untuk dapat menyelesaikan skripsi ini.
3. Bapak Eko Setiawan, S.T., M.T., Ph.D. selaku dosen pembimbing dua penulis yang telah meluangkan waktunya untuk memberikan arahan dan masukan terkait penyelesaian penulisan laporan skripsi ini.
4. Bapak Achmad Basuki, S.T., M.MG., Ph.D. selaku Ketua Jurusan Teknik Informatika Universitas Brawijaya.
5. Bapak Eko Setiawan, S.T., M.T., Ph.D. selaku Ketua Program Studi Teknik Komputer Universitas Brawijaya.
6. Bapak Ibu Dosen Fakultas Ilmu Komputer Universitas Brawijaya yang telah mengajarkan ilmunya kepada penulis selama menjalani perkuliahan.
7. Seluruh teman-teman Teknik Komputer yang telah menemani hari-hari penulis selama menjalani perkuliahan dan memberikan banyak pengalaman baru serta memberikan masukan dan inspirasi terhadap penyelesaian penelitian dan penulisan laporan skripsi ini.
8. Amalia Septi Mulyani yang selalu menemani, mendukung, menghibur, memotivasi dan menyemangati penulis dalam menyelesaikan penelitian dan penulisan laporan skripsi ini.

9. Sahabat penulis khususnya Iqbal Ridho, Alfa Resya, Riza Muztauf, Bayulistio, dan Panji Octo yang selalu menemani penulis dalam menyelesaikan penulisan laporan skripsi ini dengan canda tawa.

Penulis menyadari bahwa penelitian dan penulisan skripsi ini masih terdapat banyak kekurangan dan ketidaksempurnaan. Oleh karena itu, penulis sangat membutuhkan kritik dan saran yang membangun dari semua pihak agar penelitian ini dapat bermanfaat.

Malang, 28 Februari 2021

Munammad Bilal
mhmmdbilal246@gmail.com

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

ABSTRAK

Muhammad Bilal, implementasi Sistem Pendeteksi Fibrilasi Atrium Berdasarkan Interval dan Gradien QRS Menggunakan Metode Jaringan Saraf Tiruan

Pembimbing: Rizal Maulana, S.T., M.T., M.Sc. dan Eko Setiawan, S.T., M.T., Ph.D.

Penyakit jantung merupakan salah satu jenis penyakit kardiovaskular yang dapat menyebabkan kematian. Penyakit jantung yang paling umum terjadi adalah penyakit jantung koroner, aritmia, gagal jantung, katup jantung dan otot jantung. Sebanyak 87% penderita penyakit jantung koroner mengalami kematian mendadak akibat aritmia. Gejala umum dari penderita aritmia adalah jantung yang berdebar dengan tidak normal. Di sisi lain dari 41% pasien yang mengeluh berbedar ternyata menderita aritmia. Aritmia sendiri dapat dibedakan menjadi beberapa jenis, yaitu PAC, PVC, Takikardia Supraventrikular, Takikardia Ventrikel, Fibrilasi Atrium, Ventrikel Fibrilasi dan Bradiaritmia. Fibrilasi Atrium sendiri merupakan kerusakan irama jantung dimana sinyal listrik pada atrium tidak sesuai dengan ventrikel dalam melakukan kontraksi. Fibrilasi Atrium dapat menjadi pemicu komplikasi dengan penyakit lain, seperti kardiomiopati, hipertiroid, strok, palpitasi dan gagal jantung. Untuk mencegah hal tersebut, maka dilakukan penelitian untuk mendeteksi Fibrilasi Atrium sedini mungkin. Penelitian tersebut dilakukan dengan membuat sistem yang dibangun dari Arduino Uno sebagai pemroses mikrokontroller, sensor AD8232 sebagai perekam sinyal EKG dan LCD sebagai layar yang menampilkan informasi hasil diagnosis kondisi "Normal" atau "Fibrilasi Atrium". Selain itu sistem yang dibangun menggunakan fitur mean interval QRS, median interval QRS, mean gradient QRS dan median gradient QRS untuk melakukan klasifikasi terhadap kondisi tersebut dengan menggunakan metode klasifikasi Jaringan Saraf Tiruan. Sebanyak 40 data latih akan digunakan pada fase pelatihan metode Jaringan Saraf Tiruan untuk menghasilkan bobot dan bias tetap. Selanjutnya dilakukan pengujian untuk menguji model yang dihasilkan dari fase pelatihan. Di sisi lain pengujian BPM dilakukan untuk menguji kehandalan sensor dalam mengakuisisi sinyal EKG. Dari 10 kali pengujian BPM yang dilakukan diperoleh hasil berupa akurasi sebesar 94,55%. Kemudian dari 20 data yang diujikan pada model Jaringan Saraf Tiruan diperolen hasil berupa akurasi sebesar 90% dengan waktu komputasi selama 32,09 ms.

Kata Kunci: Fibrilasi Atrium, Interval QRS, Gradien QRS, Jaringan Saraf Tiruan.

Repository Universitas Brawijaya

ABSTRACT

Muhammad Bilal, implementation of Atrial Fibrillation Detection System Based on QRS Intervals and Gradients Using the Neural Network Method

Supervisor: Rizal Maulana, S.T., M.T., M.Sc. dan Eko Setiawan, S.T., M.T., Ph.D.

Heart disease is a type of cardiovascular disease that can cause death. The most common heart diseases are coronary heart disease, arrhythmias, heart failure, heart valves and heart muscle. 87% of people with coronary heart disease experience sudden death due to arrhythmias. The common symptom of arrhythmia sufferers is abnormal palpitations. On the other hand, 41% of patients who complained of palpitations suffered from arrhythmias. Arrhythmias can be divided into several types, including PAC, PVC, Supraventricular Tachycardia, Ventricular Tachycardia, Atrial Fibrillation, Ventricular Fibrillation and Bradyarrhythmias. Atrial fibrillation is a breakdown of the heart rhythm where the electrical signal in the atria does not match the ventricles to contract. Atrial fibrillation can lead to complications with other diseases, such as cardiomyopathy, hyperthyroidism, stroke and heart failure. To prevent this, a study was conducted to detect atrial fibrillation as early as possible. The research was carried out by making a system built from Arduino Uno as a microcontroller processor, AD8232 sensor as an ECG signal recorder and the LCD as a screen that displays information on the diagnosis of "Normal" or "Atrial Fibrillation" conditions. In addition, the system built uses the mean QRS interval, QRS median interval, mean QRS gradient and median QRS gradient to classify these conditions using the Artificial Neural Network classification method. A total of 40 training data will be used in the training phase of the Neural Network method to produce fixed weights and bias. Furthermore, testing is carried out to test the resulting model from the training phase. Then the BPM test is carried out to test the reliability of the sensor in acquiring the ECG signal. BPM testing was carried out 10 times with an accuracy rate of 94,55%. Furthermore, 20 data were tested on the Artificial Neural Network model with an accuracy rate of 90% with a computation time of 32,09 ms.

Keywords: Atrial Fibrillation, QRS Interval, QRS Gradient, Artificial Neural Network.

DAFTAR ISI

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

DAFTAR TABEL

DAFTAR GAMBAR

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

DAFTAR LAMPIRAN

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Penyakit kardiovaskular merupakan penyakit paling berbahaya di seluruh dunia yang menempati urutan nomor satu untuk penyakit yang menyebabkan kematian di dunia. Menurut data dari *World Health Organization (WHO)* terdapat hampir 18 juta orang kehilangan nyawa akibat dari penyakit tersebut setiap tahunnya dan diperkirakan akan selalu meningkat setiap tahun. Penyakit jantung dan stroke termasuk jenis penyakit kardiovaskular yang dapat menyerang berbagai kelompok usia, tidak hanya orang-orang yang sudah berusia lanjut tetapi kelompok usia produktif juga dapat mengidap penyakit jantung. Selain itu kasus dan kematian akibat dari penyakit kardiovaskular juga lebih banyak terjadi di negara berkembang karena penghasilan penduduknya rendah sampai sedang (Firdaus, 2019). Penyakit jantung koroner, otot jantung, aritmia (gangguan irama jantung), gagal jantung dan katup jantung merupakan tipe penyakit jantung yang sering ditemui (Zainury, R., 2020). Penyakit jantung dapat dihindari dengan cara melakukan pengecekan deteksi gejalanya sejak dini. Pendeteksian penyakit jantung saat ini hanya dapat dilakukan di rumah sakit untuk mendapatkan hasil dengan akurasi tinggi. Pendeteksian tersebut dilakukan dengan menggunakan beberapa cara, antara lain EKG (Elektrokardiograf), monitor *Holter* dan pemeriksaan elektrofisiologi dengan bantuan tenaga medisx.

Menurut data pasien yang menderita penyakit jantung koroner di Indonesia, sekitar 87% penderitanya meninggal dunia mendadak akibat aritmia. Kematian mendadak tersebut dapat dicegah apabila penderita melakukan pemeriksaan sejak dini secara rutin. Namun sayangnya pemahaman masyarakat mengenai aritmia sangat rendah sehingga tidak dapat mengenali sejak dini dan terlambat untuk melakukan penanganan. Padahal gejala umum dari penderita aritmia salah satunya adalah jantung yang berdebar secara tidak normal. Hal tersebut dipastikan dari pasien yang mengeluh berdebar ternyata 41% diantaranya mengidap aritmia (Yuniadi, 2017). Aritmia diklasifikasikan menjadi beberapa jenis, seperti PAC (*Premature Atrial Contraction*), PVC (*Premature Ventricular Contraction*), VT (*Ventricular Tachycardia*), AF (*Atrial Fibrillation*), SVT (*Supraventricular Tachycaraia*), VF (*Ventricular Fibrillation*) dan Bradiaritmia (Fern, 2018). Di antara berbagai jenis aritmia, penelitian ini akan difokuskan pada aritmia jenis Fibrilasi Atrium (FA) karena FA sangat erat kaitannya dengan penyakit lain, seperti hipertensi, hipertiroidisme, diabetes, dan penyakit jantung rematik sehingga amat penting untuk dideteksi sejak dini. Selain itu FA juga dapat memicu komplikasi jika tidak ditangani sejak dini, di antaranya kardiomiopati, strok, palpitasi sampai yang paling parah adalah gagal jantung (Eliana, 2019).

Fibrilasi atrium (FA) merupakan kerusakan irama detak jantung sehingga berdebar lebih kencang dan cenderung berefek pada komplikasi lain seperti mengalami gagal jantung. Penderita FA memiliki gangguan sinyal listrik yang mengatur atrium sehingga tidak melakukan kontraksi pada waktu yang sesuai

dengan ventrikel (Yuniadi, 2017). Penelitian terkait pendeteksian aritmia jenis fibrilasi atrium telah dilakukan (Sofiana, 2020). Pada penelitian tersebut menggunakan tiga fitur sebagai parameter untuk menentukan kondisi fibrilasi atrium. Tiga fitur tersebut antara lain BPM (*Beat Per Minute*), *mean* Interval RR dan *median* Interval RR. Untuk mendapatkan fitur tersebut, sinyal EKG diakusisi melalui sensor AD8232. Pada proses akuisisi sinyal EKG, digunakan filter median untuk mendeteksi titik R. Kemudian setiap titik R akan disimpan sehingga didapatkan nilai interval RR dari titik R pada waktu saat ini dan titik R pada waktu sebelumnya. Nilai interval RR akan disimpan sebanyak 8 siklus sehingga fitur mean dan median interval RR dapat ditemukan. Setelah mendapatkan nilai fitur, dilakukan klasifikasi dengan menggunakan metode SVM (*Support Vector Machine*). Metode SVM akan menentukan masukan yang diperoleh dari nilai fitur termasuk ke dalam kelas "Normal" ataupun kelas "AF". Hasil dari penelitian ini berupa tingkat akurasi dalam menghitung BPM sebesar 95,42% dan tingkat akurasi klasifikasi metode SVM sebesar 83,88% dari 12 data yang diujikan. Selain itu, didapatkan juga waktu komputasi dari fase *training* dan testing berturut-turut adalah 219,30 ms dan 0,09 ms.

Pada penelitian serupa, metode lain juga dapat digunakan untuk mengklasifikasikan FA, salah satunya adalah metode Jaringan Saraf Tiruan (JST). Dari penelitian yang dilakukan oleh (Macknickas, 2017) metode JST dapat dikombinasikan dengan fitur yang cukup banyak yaitu 23 fitur, selain itu hasil yang didapatkan juga cukup baik dengan akurasi sebesar 78%. Penelitian yang dilakukan oleh (Anzihory, 2016) juga menggunakan metode JST sebagai klasifikasinya dengan hasil tingkat akurasi diatas 90%. Oleh karena itu, pada penelitian ini akan digunakan JST sebagai metode klasifikasi penyakit FA. Nantinya fitur yang didapatkan melalui hasil sinyal EKG digunakan sebagai masukan JST untuk menentukan kondisi FA atau normal. Selain interval RR fitur lain juga dapat digunakan untuk mendeteksi penyakit FA, yaitu melalui sinyal kompleks QRS (Macknickas, 2017). Beberapa fitur yang dapat digunakan dari sinyal kompleks QRS antara lain interval QRS dan gradien QRS. Hasil sinyal EKG pada penderita FA biasanya terdapat kompleks QRS yang ireguler dimana kompleks QRS tersebut lebih lebar dari kondisi normal (Yuniadi, 2014). Secara otomatis hal tersebut akan menghasilkan perbedaan nilai interval maupun nilai gradien QRS antara kondisi FA dan kondisi normal. Selain itu, pada penelitian sebelumnya, kedua fitur tersebut telah terbukti mampu mendeteksi penyakit terkait jantung, seperti *atrial fibrillation* (Macknickas, 2017) dan *sleep apnea* (Indrawati, 2020). Kedua fitur tersebut juga memiliki kelebihan karena dalam perhitungannya cukup dengan menggunakan kalkulasi matematika sederhana.

Oleh karena itu, dalam penelitian ini akan dibangun sistem yang dapat mendiagnosis penyakit fibrilasi atrium (FA) berdasarkan empat fitur (Rata-rata Interval QRS, Modus Interval QRS, Rata-rata Gradien QRS dan Modus Gradien QRS) yang didapatkan dari pendeteksian sinyal EKG dengan menggunakan sensor yang mampu merekam aktivitas listrik pada jantung, yaitu sensor AD8232. Dengan nilai rata-rata dan nilai median dari fitur yang digunakan maka sistem akan mendeteksi irama jantung secara akurat, karena pendeteksian dilakukan secara keseluruhan

tidak hanya pada satu titik rekanam irama jantung saja. Sensor AD8232 akan menangkap sinyal EKG melalui tiga elektrode yang ditempatkan ke bagian tubuh tertentu sinyal biopotensial melalui tiga buah elektroda yang ditempatkan pada bagian tubuh tertentu, yang masing-masing berfungsi sebagai elektrode positif, elektrode negated serta referensi (Hariri, 2019). Alasan penggunaan sensor AD8232 adalah karena penelitian yang dilakukan menurut (Hariri, 2019) dan (Sofiana, 2020) membuktikan bahwa sensor tersebut dapat mendeteksi dengan baik. Kemudian untuk melakukan analisis terhadap sinyal EKG yang diakuisisi oleh sensor AD8232, Arduino Uno akan mengolahnya untuk mendapatkan nilai fitur yang digunakan dan kemudian diklasifikasikan dengan metode pengklasifikasi Jaringan Saraf Tiruan. Metode jaringan saraf tiruan selanjutnya akan dihitung kecepatannya dalam mendeteksi penyakit FA untuk memastikan sistem tidak hanya dapat mendeteksi dengan akurat, tetapi juga mengetahui waktu komputasi yang dibutuhkan sistem dalam mendeteksi penyakit tersebut.

Berdasarkan fakta di atas, maka penelitian ini akan berfokus pada deteksi penyakit aritmia jenis Fibrilasi Atrium (FA) dengan menganalisis hasil sinyal EKG yang diperoleh melalui sensor AD8232 dan kemudian diolah melalui Arduino Uno yang mengimplementasikan metode Jaringan Saraf Tiruan. Sehingga secara garis besar, sistem ditujukan untuk menyelesaikan masalah berupa pendeteksian penyakit FA. Selain itu, masalah lain seperti waktu dan biaya juga diharapkan dapat ditekan dengan adanya sistem ini. Kemudian untuk proses mengakuisisi data, mengolah data serta langkah dalam membangun sistem akan dijelaskan dengan detail pada bahasan-bahasan selanjutnya.

1.2 Rumusan Masalah

Pada sub bab ini merupakan rumusan masalah yang dihadapi dari latar belakang penelitian ini. Rumusan masalah tersebut, antara lain:

1. Bagaimana akurasi pembacaan nilai sensor AD8232 dalam mengakuisisi sinyal EKG pada jantung?
2. Bagiamana tingkat akurasi dalam mendeteksi kondisi normal maupun kondisi fibrilasi atrium pada sistem yang mengimplementasikan metode jaringan saraf tiruan?
3. Bagiamana waktu komputasi metode jaringan saraf tiruan dalam menentukan kondisi normal maupun kondisi fibrilasi atrium?

1.3 Tujuan

Pada sub bab ini merupakan tujuan yang ingin dicapai dari penelitian ini dan merupakan jawaban dari rumusan masalah yang muncul dari latar belakang. Tujuan dari penelitian ini diantaranya:

1. Menguji akurasi pembacaan sensor AD8232 dalam mendeteksi sinyal EKG pada subjek.

2. Menguji metode Jaringan Saraf Tiruan yang diimplementasikan untuk mengetahui keakuratan dalam membedakan kondisi Fibrilasi Atrium dan kondisi normal.
3. Menguji waktu komputasi metode jaringan saraf tiruan dalam menentukan penyakit fibrilasi atrium maupun kondisi normal.

1.4 Manfaat

Manfaat dari penelitian ini adalah membantu masyarakat untuk dapat mendeteksi penyakit jantung sejak dini khususnya dalam mendiagnosis penyakit Fibrilasi Atrium. Dengan sistem yang dihasilkan dari penelitian ini, masyarakat dapat mengeluarkan biaya yang lebih sedikit namun tidak mengurangi kualitas dari hasil pendeteksian yang dilakukan.

1.5 Batasan Masalah

Untuk mendapatkan hasil yang maksimal, maka dari rumusan masalah yang ada, aspek yang mempengaruhi pengelitian ini akan dibatasi sehingga penelitian tetap berfokus pada rumusan masalah yang ada. Masalah yang akan dibatasi antara lain:

1. Kondisi yang dapat dideteksi terbagi menjadi kondisi normal dan kondisi penyakit Fibrilasi Atirum.
2. Sensor yang digunakan hanya dapat mendeteksi aktivitas listrik saja yang ada pada jantung.
3. Jumlah elektroda yang digunakan adalah sebanyak 3 buah elektrode.
4. Data latih yang digunakan adalah sebanyak 40 data dimana 20 diantaranya adalah data kondisi normal dan 20 lainnya adalah data kondisi fibrilasi atrium.
5. Data uji berjumlah 20 data dimana 10 diantaranya adalah kondisi fibrilasi atrium dan 10 lainnya adalah kondisi normal..

1.6 Sistematika Pembahasan

Secara struktural, penulisan laporan pada penelitian ini dibagi menjadi tujuh Bab bahasan sebagai berikut.

BAB I PENDAHULUAN

Bab ini berisi tentang latar belakang penelitian, rumusan masalah, tujuan, manfaat, batasan masalah dan sistematika penulisan penelitian ini.

BAB II LANDASAN KEPUSTAKAAN

Bab ini berisi tentang kajian pustaka dan teori-teori pendukung tentang implementasi sistem pendeteksi fibrilasi atrium menggunakan metode jaringan saraf tiruan.

BAB III METODOLOGI PENELITIAN

Bab ini berisi tentang metode penelitian yang digunakan pada implementasi sistem pendeteksi fibrilasi atrium menggunakan metode jaringan saraf tiruan.

BAB IV REKAYASA KEBUTUHAN

Bab ini berisi tentang kebutuhan yang diperlukan oleh sistem yang akan dibangun pada penelitian ini.

BAB V PERANCANGAN DAN IMPLEMENTASI

Bab ini berisi tentang perancangan dan implementasi dari kebutuhan sistem yang telah dijelaskan pada bab sebelumnya.

BAB VI PENGUJIAN

Bab ini berisi tentang pelaksanaan pengujian dan analisis pada implementasi sistem pendeteksi fibrilasi atrium menggunakan metode jaringan saraf tiruan.

BAB VII PENUTUP

Bab ini berisi tentang kesimpulan dari penelitian yang dilakukan serta saran untuk pengembangan selanjutnya dari pelaksanaan penelitian ini.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Landasan kepustakaan berisi tentang kajian pustaka yang bertujuan untuk mendukung proses penelitian. Kajian pustaka dapat memberikan bantuan dan infrmasi untuk mempersiapkan penelitian. Oleh karena itu, pencarian dilakukan dari penelitian serupa yang dillakukan sebelumnya. Tabel 2.1 merepresentasikan penelitian serupa sebelumnya.

Tabel 2.1 Kajian Pustaka

No.	Judul Penelitian	Peneliti	Persamaan	Perbedaan
1	*Atrial Fibrillation Classification Using QRS Complex Features and LSTM.* (2017)	Maknickas, V., et al.	Klasifikasi atrial fibrilasi menggunakan kompleks QRS.	Metode untuk klasifikasi menggunakan jaringan saraf jenis LSTM.
2.	Deteksi Fibrilasi Atrium Menggunakan FAM yang Dikombinasikan dengan *Grid Search* (2019)	Eliana, M., et al.	Eliana, M., et al. membangun sistem pendeteksi atrial fibrilasi.	Menggunakan FAM dan *Grid Search* untuk mendeteksi atrial fibrilasi.
3.	Klasifikasi Aritmia dari Hasil Elektrokardiogram Menggunakan *Support Vector Machine* dengan Seleksi Fitur Menggunakan Algoritma Genetika. (2018)	Cahya, R. A., et al.	Klasifikasi aritmia menggunakan hasil sinyal EKG.	Metode untuk klasifikasi menggunakan SVM dan fitur algoritma genetika.
4.	Sistem Deteksi Fibrilasi Atrium menggunakan Fitur RR Elektrokardiogram dengan Jaringan Saraf Tiruan. (2016)	Anzihory, E., et al.	Anzihory, E., et al. membuat sistem pendeteksi atrial fibrilasi dengan metode jaringan saraf tiruan.	Fitur yang digunakan untuk mendeteksi penyakit atrial fibrilasi adalah interval RR.
5	Sistem Deteksi Penderita Aritmia Berdasarkan Jumlah Detak Jantung Berbasis Smartphone. (2017)	Gustini, E., et al.	Gustini, E., et al. membuat sistem pendeteksi aritmia.	Menggunakan sensor PPG untuk mendeteksi hasil sinyal EKG.

Dalam penelitian yang dilakukan oleh Maknickas *et. al* (2017) dibuat sistem pendeteksi penyakit aritmia jenis fibrilasi atrium. Sistem tersebut mengklasifikasikan sinyal EKG menjadi empat kelas, yaitu Normal, *Atrial* Fibrillation, *Others* dan *Too Noisy* dengan metode jaringan saraf LSTM (*Long Short*

Term Memory). Dalam prosesnya, sistem tersebut menggunakan hasil sinyal EKG yang selanjutnya akan dilakukan filter dengan tujuan untuk menghilangkan *noise*nya serta mendeteksi puncak R. Setelah puncak R terdeteksi, barulah kemudian dilakukan ekstraksi fitur. Ekstraksi fitur yang digunakan pada penelitian ini sebanyak 23 fitur, diantaranya nilai interval, nilai puncak, panjang interval, simpangan baku, amplitudo dan panjang sinyal hasil EKG. Kemudian dari seluruh sampel data dibagi menjadi 80% data latih dan 20% data uji. Hasil dari sistem yang dibangun pada penelitian ini adalah mendapatkan akurasi sebesar 78%. Kelebihan dari penelitian ini adalah menggunakan fitur yang sangat banyak yaitu 23 fitur. Kekurangan dari sistem ini adalah meskipun sudah menggunakan fitur yang cukup banyak namun menghasilkan akuras yang tidak terlalu tinggi secara keseluruhannya.

Eliana *et. al* (2019) melakukan sebuan penelitian yang membangun sistem pendeteksi fibrilasi atrium dengan metode *Finding Anomalies Arround the Mean* (FAM) dan *Grid Search*. Metodologi pada penelitian ini terbagi atas empat tahap, yaitu persiapan data, perancangan fitur FAM, pencarian nilai parameter dengan *Grid Search* dan penerapan pada data uji. Data pada penelitian ini bersumber dari *PhysioNet* dengan mengambil hasil sinyal EKG dari 22 pasien. Kemudian data tersebut disiapkan dengan mencari puncak R, membuat interval RR dan melakukan segmentasi. Pada penelitian ini menggunakan dua parameter dan akan dideteksi *irreguluarity* pada data yang disiapkan dengan cara menemukan rata-rata panjang interval, nilai ambang batas, menentukan interval tak normal dan nilai status segmen. *Grid search* kemudian digunakan untuk mencari nilai optimal dari parameter yang digunakan. Hasil yang didapatkan dari penelitian ini adalah berupa nilai akurasi sebesar 92,63%, nilai sensitivitas 95,37% dan nilai spesifisitas 90,58%. Kelebihan dari penelitian ini adalah meskipun tidak menggunakan metode klasifikasi ataupun klastering namun dapat menghasilkan nilai akurasi yang tinggi. Kelemahan dari penelitian ini adalah parameter yang digunakan untuk mendeteksi penyakit atrial fibrilasi terlalu sedikit yaitu hanya 2 buah parameter.

Dalam penelitian lain yang dilakukan oleh Cahya *et. al* (2017) menggunakan metode *Support Vector Machine* (SVM) untuk membuat sistem klasifikasi aritmia dari hasil EKG dengan algoritma genetika untuk memilih fitur dan dataset. Algoritma tersebut digunakan untuk mendapatkan jumlah fitur yang lebih sedikit nantinya. Sedangkan untuk dataset yang telah dipilih akan digunakan sebagai data latih pada SVM untuk dapat mengklasifikasikan aritmia. Dari pengujian yang dilakukan, didapatkan hasil akurasi keselurhan 82,5% dari 140 data yang ada meliputi 120 training set data dan 20 testing set data. Fitur dapat dikerucutkan menjadi 406 fitur yang semula adalah 2160 fitur berkat Algoritma genetika-SVM. Kelebihan penelitian ini adalah menggunakan parameter yang terbilang cukup banyak dari algoritma genetika-SVM untuk pengujian guna mengetahui dampak dari parameter tersebut terhadap keluaran sistem. Di sisi lain, sistem ini memiliki kelemahan yaitu untuk setiap parameter, algoritma genetika SVM harus diuji 10 kali untuk mendapatkan nilai fitness rata-rata dan hasil yang lebih stabil.

Penelitian yang serupa juga dilakukan oleh Anzihory *et. al* (2016). Penelitian dilakukan dengan memanfaatkan hasil sinyal EKG yang digunakan sebagai informasi masukan untuk metode klasifikasi jaringan saraf tiruan. Pada penelitian ini pengambilan sampel dilakukan melalui basis data *Massachusetts Institute of Technology Beth Israel Hospital* (MIT-BIH) sebanyak 17 data pasien. Kemudian fitur yang digunakan dalam penelitian ini adalah 7 fitur pada statistik RR diantaranya *mean*, *modus*, *median*, maksimum, minimum, jangkauan dan standar deviasi dari sinyal RR. Kemudian dari data yang diperoleh sebanyak 10% digunakan sebagai data latih dan 90% digunakan sebagai data uji. Metode jaringan saraf yang digunakan memiliki tiga variasi, yaitu *Learning Vector Quantization* (LVQ), *Radial Basis Function* (RBF) dan *Multilayer Perception* (MLP). Dari pengujian yang dilakukan, diperoleh hasil metode RBF dengan 99,97% Sensitivitas, 99,84% Spesifisitas dan 99,89% Akurasi sebagai metode jaringan saraf tiruan yang paling baik. Kelebihan dari penelitian ini adalah mencoba beberapa metode jaringan saraf tiruan untuk menemukan metode terbaik dalam klasifikasi penyakit atrial fibrilasi. Kekurangan dari penelitian ini adalah pembagian antara data latih dan data uji yang tidak sesuai dengan kaidah dalam menentukan klasifikasi suatu objek.

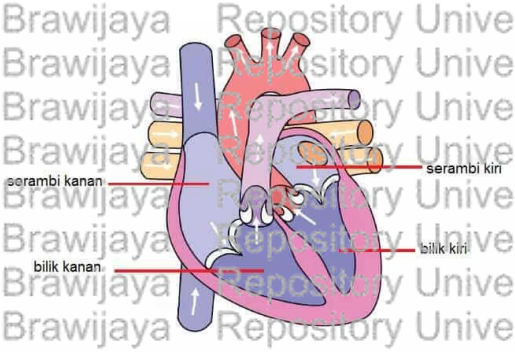
Gustini et. al (2017) melakukan penelitian yang serupa yaitu membuat sistem deteksi penderita aritmia berdasarkan jumlah detak jantung berbasis *smartphone*. Pada penelitian ini untuk mendeteksi detak jantung digunakan pulse sensor yang terdiri dari LED dan sensor cahaya dimana akan menghasilkan data analog sinyal *Photoplethysmograph* (PPG) nantinya. Pengujian detak jantung dari pulse sensor yang berupa data analog tadi akan diolah menjadi ADC pada mikrokontroler Arduino uno. Hasil dari pengujian data ADC nantinya akan dikirim ke *smartphone* menggunakan modul *Bluetooth HC-05*. Hasilnya perbedaan detak jantung untuk penderita aritma adalah 44 bpm (bradikardia) dan 194 bpm (takikardia). Kelebihan dari penelitian ini adalah sistem yang dibuat sangat memperhatikan *portability* sehingga dapat digunakan secara mandiri dan tidak terbatas tempat. Kekurangan pada sistem ini adalah harus dilakukan 3 kali pengambilan data untuk mendapatkan nilai yang stabil.

Pada penelitian sebelumnya sesuai dengan Tabel 2.1, terdapat banyak parameter dan fitur yang digunakan untuk mendeteksi Fibrilasi Atrium (FA). Selain itu diperlukan tahapan-tahapan sebelum dan saat data di proses untuk melakukan pengklasifikasiannya. Pengambilan data juga dilakukan lebih dari satu kali untuk mendapatkan hasil yang stabil. Untuk menghindari kekurangan tersebut, pada penelitian ini jumlah fitur yang digunakan disesuaikan agar tidak terlampau sedikit namun juga tidak terlalu banyak untuk menjaga tingkat akurasi sistem itu sendiri. Kemudian pembagian data latih dan data uji pada dataset yang digunakan adalah 2 banding 1 agar tetap sesuai dengan kaidah pembagian dataset yang seharusnya. Sementara itu dari kelebihan yang terdapat pada penelitian sebelumnya, fitur statistik akan digunakan untuk melakukan pengklasifikasian terhadap sinyal EKG yang dideteksi dan menggunakan salah satu metode genetika yaitu Jaringan Saraf Tiruan sebagai metode klasifikasi untuk menentukan kondisi normal maupun kondisi FA.

2.2 Dasar Teori

2.2.1 Jantung

Salah satu organ tubuh manusia yang paling penting adalah jantung. Dimana jantung merupakan bagian dari sistem kardiovaskular manusia yang berperan sebagai penggerak dalam pengangkutan nutrisi dan oksigen ke seluruh tubuh. Fungsi utama jantung adalah memompa darah ke seluruh tubuh. Jantung memiliki empat rongga. Atrium kanan (serambi kanan) dan atrium kiri (serambi kiri) merupakan dua rongga yang terdapat pada bagian atas jantung. Sedangkan ventrikel kanan (bilik kanan) dan ventrikel kiri (bilik kiri) merupakan dua rongga yang terdapat pada bagian bawah jantung (Kurniawan, 2019). Gambar 2.1 menunjukkan ilustrasi dari jantung beserta keempat rongga yang terdapat pada jantung itu sendiri.



Gambar 2.1 Jantung

Sumber : (https://kependidikan.com/)

Keempat bagian jantung tersebut memiliki fungsi masing-masing, yaitu:

1. Atrium Kanan (serambi kanan)

   Atrium Kanan (serambi kanan) berfungsi sebagai tempat kembalinya darah dari seluruh tubuh yang mengandung karbondioksida yang nantinya akan dibawa ke ventrikel kanan.

2. Atrium kiri (serambi kiri)

   Atrium kiri (serambi kiri) berfungsi sebagai tempat kembalinya daran dari paru-paru yang mengandung oksigen yang nantinya akan dibawa ke ventrikel kiri.

3. Ventrikel Kanan (bilik kanan)

   Ventrikel kanan (bilik kanan) berfungsi sebagai rongga yang menerima darah dari atrium dari atrium kanan yang nantinya akan dibawa ke paru-paru.

4. Ventrikel Kiri (bilik kiri)

   Ventrikel kiri (bilik kiri) berfungsi sebagai rongga yang menerima darah dari atrium kiri yang nantinya dibawa ke seluruh tubuh.

2.2.2 Aritmia (Gangguan Irama Jantung)

Aritmia merupakan keadaan detak jantung yang tidak normal dengan gejala detak jantung menjadi tidak teratur, terlalu cepat atau terlalu lambat. Keadaan tersebut dapat membawa penderita dalam masalah kardiovaskular lainnya, seperti hipertensi, katup jantung, arteri koroner sampai yang paling parah adalah gagal jantung (Lim, 2019). Menurut Bazudewa *et. al* (2020) secara umum penyakit aritmia dibagi menjadi dua, yaitu:

1. Bradikardia

   Bradikardia merupakan kondisi dimana detak jantung terjadi lebih lambat dari detak jantung normal. ditandai dengan detak jantung yang kurang dari 60 kali dalam satu menit, dimana pada kondisi normal biasanya jantung berdetak sebanyak 60 sampai 100 kali dalam satu menit. Penderita bradikardia biasanya terjadi pada usia lanjut serta memiliki riwayat penyakit kardiovaskular lain, atau efek samping dari merokok dan konsumsi obat-obatan. Bradikardia juga dapat menjadi gejala adanya masalah pada sistem listrik jantung. Efek yang timbul pada penderita bradikardia biasanya keluhan seperti sesak napas, tidak mudah berkonsentrasi, mengalami pingsan, pusing kepala dan kelelahan saat hanya melakukan sedikit aktivitas.

2. Takikardia

   Takikardia merupakan kondisi dimana detak jantung terjadi lebih cepat dari detak jantung normal. ditandai dengan jantung yang berdetak lebih cepat dari yang seharusnya. Biasanya penyebab dari penderita tidak selalu dapat diketahui, namun bisa muncul karena beberapa faktor, seperti faktor keturunan, riwayat penyakit jantung dan anemia atau efek samping dari obat-obatan, kebiasaan merokok dan mengonsumsi minuman beralkohol. Takikardia biasanya dapat menimbulkan keluhan seperti nyeri dada, pusing, kelelahan dan sesak napas. Namun pada beberapa kasus juga tidak menimbulkan gejala apapun.

Secara umum penyakit aritmia memang terbagi atas dua seperti penjelasan diatas, namun secara detailnya penyakit aritmia bermacam-macam meskipun memiliki kondisi detak jantung yang serupa (Fern, 2018). Macam-macam penyakit aritmia terbagi atas:

1. PVC (*Premature Ventrikel Contraction*)

   PVC (*Premature Ventrikel Contraction*) merupakan kondisi detak jantung cepat yang berasal dari salah satu ventrikel. Kondisi ini menyebabkan ritme jantung penderita menjadi terganggu dan penderita merasa berdebar atau detak jantung yang terlewat. Kondisi ini terjadi saat kontraksi ventrikel terjadi lebih cepat dari detak jantung normal sehingga mengganggu siklusnya (Mayoclinic, 2019).

2. PAC (*Premature Atrial Contraction*)

PAC (*Premature Atrial Contraction*) merupakan kondisi detak jantung yang lebih cepat dan tidak normal yang terdapat pada atrium serta dapat mengganggu ritme jantung. Penderita biasanya tidak mengalami gejala saat didiagnos. Namun penderita yang mengalami gejala biasanya merasakan jantungnya berdebar yang muncul saat malam hari. Selain itu penderita juga memiliki gejala pusing atau nyeri dada (Sullivan, D., 2017).

3. Takikardia Supraventrikular

Takikardia Supraventrikular merupakan kondisi dimana denyut jantung yang cepat karena impuls listrik pada atrioventricular (bagian sistem control listrik jantung) tidak sesuai siklus. Kondisi ini dapat menyebabkan denyut per menit setidaknya 100 kali berdetak atau bahkan mencapai 300 kali. Gejalanya ditandai dengan jantung yang berdebar kencang, berkeringat atau merasa pusing. Selain disebabkan oleh sistem listrik pada jantung, penyebab lain juga dapat memicu kondisi ini, seperti efek samping obat-obatan, kondisi kesehatan serta keturunan (Nhs, 2018).

4. Takikardia Ventrikel

Takikardia ventrikular merupakan kondisi detak jantung yang cepat karena kegagalan fungsi sistem listrik yang ada pada jantung. Gejalanya ditandai dengan jantung yang berdebar, nyeri dada dan kecemasan. Kondisi ini paling umum terjadi pada orang yang memiliki kerusakan pada otot jantung sehingga menciptakan jalur listrik yang tidak normal pada ventrikel. Kondisi ini juga dapat menimbulkan risiko penyakit lainnya seperti penyakit arteri koroner dan penyempitan pembuluh darah (Setiawan, d., 2019).

5. Fibrilasi Atrium

Fibrilasi atrium (FA) merupakan jenis aritmia paling umum yang menyebabkan detak jantung tidak teratur serta meningkatkan resiko penyakit stroke. Pada kondisi ini ruang jantung bagian atas mengalami aktivitas listrik yang tidak normal sehingga mengacaukan ventrikel berdetak tidak teratur dan mengacaukan ritme jantung (Utari, 2019).

6. Fibrilasi Ventrikel

Fibrilasi ventrikel merupakan kondisi ketika jantung berdetak dengan impuls listrik yang cepat dan tidak menentu sehingga ventrikel hanya bergetar dan tidak memompa darah ke seluruh tubuh. Kondisi ini terjadi akibat gangguan aliran listrik sesaat setelah mengalami serangan jantung. Gejalanya ditandai dengan detak jantung yang sangat cepat, nyeri dada, pusing dan hilangnya kesadaran (Halodoc, 2020).

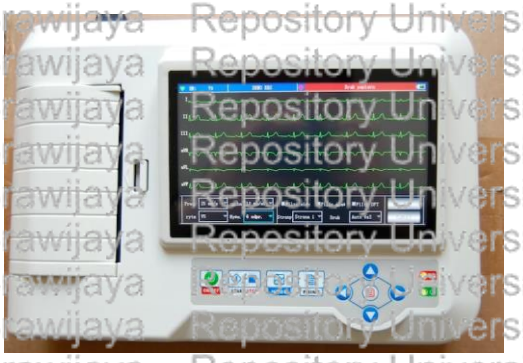
7. Bradiaritmia

Bradiaritmia merupakan kondisi dimana jantung berdetak lebih lambat dari yang seharusnya akibat dari gangguan aktivitas listrik pada jantung. Kondisi ini dapat menyebabkan anggota tubuh kekurangan jumlah darah. Gejalanya ditandai dengan pusing, sesak napas, nyeri dada, pingsan dan mudah lelah (Halodoc, 2020).

Dari macam-macam jenis aritmia diatas, setiap jenis aritmia memiliki perbedaan yang dapat dilihat melalui sinyal EKG yang dihasilkan. Seperti pada penderita PVC dengan penderita Fibrilasi Atrium, dimana penderita PVC memiliki masalah pada jantung bagian bawah atau disebut ventrikel, sementara penderita Fibrilasi Atrium memiliki masalah pada jantung bagian atas atau disebut atrium. Selain itu, pada hasil sinyal EKG yang dihasilkan, kondisi Fibrilasi Atrium pada umumnya tidak memiliki gelombang P, QRS kompleks terjadi dibawah 120 ms, gelombang ST yang lebih lebar dan munculnya fibrilasi atau disebut (*f-waves*) yang berbentuk seperti gerigi (Burns, Ed., 2020). Sedangkan pada kondisi PVC umumnya terbagi menjadi dua macam, yaitu *bigeminy* dan *trigeminy*. Kondisi *bigeminy* terjadi saat sinyal PVC muncul setelah satu siklus detak jantung, sementara kondisi *trigeminy* terjadi saat sinyal PVC muncul setelah dua siklus detak jantung (Gilang, 2018).

2.2.3 EKG (Elektrokardiogram)

EKG (elektrokardiogram) merupakan alat yang digunakan untuk melihat potensial listrik yang melewati jantung (Ernawati, 2017). Cara kerja EKG adalah memasangkan 3 elektroda pada titik *Lead* yang telah ditentukan. Elektroda yang digunakan adalah elektroda merah, elektroda kuning dan elektroda hitam. Elektroda merah dipasangkan pada pergelangan tangan kanan. Elektroda kuning dipasangkan pada pergelangan tanan kiri. Elektroda hitam dipasangkan pada pergelangan kaki kanan (Pratama, 2017). Gambar 2.2 merepresentasikan elektrokardiogram yang ditampilkan pada mesin.

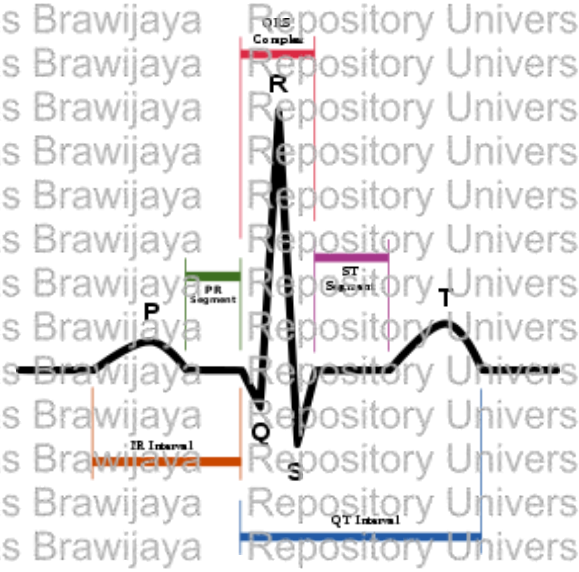


Gambar 2.2 Fektrokardiogram

Sumber: (https://www.medicalogy.com/)

2.2.4 Sinyal Elektrokardiogram

Sinyal Elektrokardiogram (EKG) merupakan representasi aktivitas jantung yang terekam oleh mesin EKG. Satu detak jantung mewakili satu siklus sinyal EKG yang terekam dengan morfologi sinyal yang terdiri dari gelombang P, kompleks QRS dan gelombang T (Utari, 2016). Bentuk normal sinyal EKG ditunjukkan pada Gambar 2.3.



Gambar 2.3 Sinyal Elektrokardiogram

Sumber : (https://id.wikipedia.org/)

Sesuai dengan Gambar 2.3, titik P merupakan gelombang bukit pertama dalam satu siklus jantung. Sementara titik Q merupakan lembah pertama setelah titik P. Kemudian titik R merupakan titik tertinggi di antara titik Q dan titik S dan dilanjutkan dengan titik S yang merupakan lembah kedua dalam satu siklus jantung. Sedangkan titik T merupakan gelombang bukit terakhir setelah titik S. Dari titik-titik tersebut akan didapatkan parameter yang akan digunakan untuk mendeteksi penyakit Fibrilasi Atrium.

$$Interval\ QRS = xS - xQ \quad (2.1)$$

$$Gradien\ QRS = \frac{yR - yQ}{xR - xQ} \quad (2.2)$$

Keterangan:

xS = data titik S terhadap sumbu horizontal

xQ = data titik Q terhadap sumbu horizontal

xR = data titik R terhadap sumbu horizontal

yR = data titik R terhadap sumbu vertikal

yQ = data titik Q terhadap sumbu vertikal

Persamaan 2.1 dan Persamaan 2.2 berturut-turut menunjukkan persamaan yang digunakan untuk mendapatkan nilai interval QRS dan gradien QRS. Nilai

interval QRS sendiri didapatkan dari mengurangkan nilai titik S dengan nilai titik Q pada sumbu horizontal. Sementara nilai gradien QRS didapatkan dari hasil selisih koordinat sumbu horizontal pada titik Q dan titik R kemudian dibagi dengan hasil selisih koordinat sumbu vertikal pada titik Q dan titik R (Indrawati, 2020).

Hasil Sinyal EKG dari aktivitas jantung juga dapat mengidentifikasi seseorang memiliki detak jantung normal atau abnormal, diantaranya:

1. Detak Jantung Normal

Hasil sinyal EKG dapat dikatakan normal jika memiliki detak jantung sekitar 60-100 kali setiap menitnya dalam kondisi rileks. Namun hal tersebut juga bergantung pada aktivitas yang biasa dilakukan. Orang yang memiliki detak jantung normal menandakan sistem listrik jantung berfungsi dengan baik. Jika diamati secara manual, detak jantung normal akan terdengar seirama, seragam dan tidak ada detak jantung tambahan (Adrian, K., 2018).

Gambar 2.4 Hasil Sinyal EKG Normal

Sumber : (https://www.ecgwaves.com/)

Pada Gambar 2.4 dapat dilihat hasil sinyal EKG dari detak jantung normal, dimana laju ventrikel berada diantara 50-100 detak jantung per menit. Selain itu juga ditandai dengan gelombang P yang terlihat konstan sebelum munculnya gelombang QRS (Rawshani, A., 2016).

2. Detak Jantung Bradikardia

Hasil sinyal EKG pada penderita bradikardia ditandai dengan irama yang lebih lambat dari irama normal, yaitu laju ventrikel berada dibawah 50 detak jantung per menit. Namun gelombang P masih dapat terlihat secara konstan sebelum gelombang kompleks QRS (Rawshani, A., 2016).

Gambar 2.5 Hasil Sinyal EKG Bradikardia

Sumber : (https://www.teachingmedicine.com/)

Pada Gambar 2.5 dapat dilihat salah satu contoh kondisi bradikardia, dimana jika dilihat secara manual pada kertas kotak pada Gambar 2.5, jarak antara Gelombang R pada siklus pertama dan kedua lebih lambat dari irama normal, kemudian untuk hitung detak jantungnya dengan cara membagi nilai 300 dengan kotak persegi yang berada diantara gelombang R tersebut, yaitu 8 kotak persegi sehingga detak jantungnya adalah 37 detak jantung per menit.

3. Detak Jantung Takikardia

Hasil sinyal EKG pada penderita takikardia merupakan kebalikan dari bradikardia, dimana irama detak jantung lebih cepat dari irama normal, yaitu laju ventrikel lebih dari 100 detak jantung per menit (Rawshani, A., 2016).

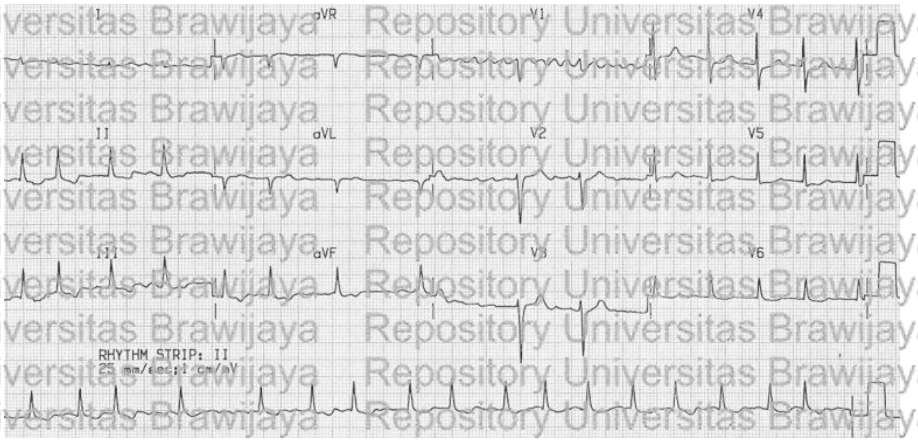
Gambar 2.6 Hasil Sinyal EKG Takikardia

Sumber : (https://www.ecgwaves.com/)

Pada Gambar 2.6 dapat dilihat salah satu contoh sinus takikardia, dimana jika diamati secara manual pada kertas kotak pada gambar, siklus detak jantung terjadi lebih cepat, kemudian untuk menghitung detak jantungnya dengan cara membagi 300 dengan kotak persegi diantara puncak R, yaitu 3 kotak persegi sehingga detak jantungnya adalah 100 detak jantung per menit.

4. Detak Jantung Fibrilasi Atrium

Hasil sinyal EKG pada penderita fibrilasi atrium pada umumnya memiliki irama yang tidak teratur, tidak memiliki gelombang P, laju ventrikel tidak konstan, kompleks QRS biasanya berada dibawah 120 ms serta muncul gelombang fibrilasi (*f-waves*) yang berbentuk seperti gerigi halus (amplitude kurang dari 50) ataupun kasar (amplitude lebih dari 50).



Gambar 2.7 Hasil Sinyal EKG Fibrilasi Atrium

Sumber : (https://www.litfl.com/)

Pada Gambar 2.7 dapat dilihat salah satu contoh penderita fibrilasi atrium, dimana terdapat gelombang fibrilasi (*f-waves*) kasar pada V1. Selain itu, pada bagian gelombang ST terlihat lebih renggang pada V6, II, III dan aVF serta respon dari ventrikel yang tidak teratur (Burns, Ed., 2020).

5. Detak Jantung *Premature Ventricular Contraction* (PVC)

Hasil sinyal EKG pada kondisi PVC pada umumnya terjadi pada 2 kondisi, yaitu setelah satu siklus detak jantung normal yang biasa disebut *bigeminy* atau setelah dua siklus detak jantung normal yang biasa disebut *trigeminy*. Kondisi PVC akan menyebabkan terlambatnya detak jantung berikutnya dan jika dibiarkan dapat menyebabkan palpitasi (Gilang, 2018). Hasil sinyal EKG dengan kondisi *bigeminy* dan *trigeminy* ditunjukkan pada Gambar 2.8.

Gambar 2.8 Hasil Sinyal EKG PVC

Sumber: (https://ecg-educator.blogspot.com/)

2.2.5 Sensor AD8232

Dalam mengolah sinyal EKG, sensor AD8232 merupakan modul *low power* yang menguatkan sinyal EKG yang akan dideteksi serta mengurangi *noise*. Modul tersebut terdiri dari dua bagian, yaitu elektroda dan Sensor EKG AD8232. Elektroda pada sensor AD3232 terdiri dari 3 buah elektroda dimana masing-masing memiliki warna merah, kuning dan hijau yang ditempatkan di dada bagian kanan (untuk warna merah), dada bagian kiri (untuk warna kuning) dan perut kanan bagian bawah (untuk warna kuning). Elektroda yang digunakan disini adalah elektroda kering sehingga aman untuk subjek penelitian. Selain itu jenis elektroda yang digunakan berbeda dari elektroda basah ataupun *reusable* elektroda sehingga lebih higienis.

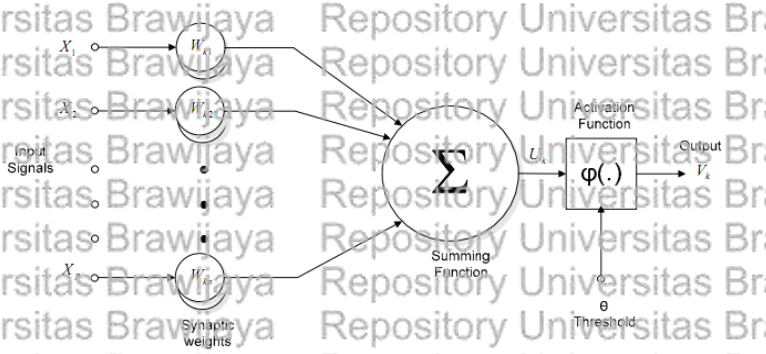
Sensor AD3232 sendiri merupakan *chip* berukuran kecil yang berfungsi untuk mengukur aktivitas listrik pada detak jantung manusia. Sensor AD8232 terdiri dari serangkaian instrumentasi *amplifier, buffer* serta sirkuit sebagai penghubung dengan elektroda. Intrumentasi *amplifier* pada sensor AD8232 dapat memperkuat sinyal EKG sekaligus mengurangi *noise* pada pembacaan sinyal EKG. Dengan arsitektur tersebut sensor AD8232 dapat mendeteksi serta menguatkan sinyal EKG dan dapat langsung dihubungkan dengan mikrokontroller (Prasad, A. S., 2019). Gambar 2.4 menunjukkan sensor AD8232 beserta elektrodenya.

Gambar 2.9 Sensor AD8232

Sumber: (https://www.techtonics.in/)

2.2.6 Metode Pengkiasifikasi Jaringan Saraf Tiruan

Metode Jaringan Saraf Tiruan (JST) adalah suatu sistem pengolah informasi yang cara kerjanya serupa dengan sistem neuron pada manusia dan digunakan dalam klasifikasi suatu objek atau dalam pengenalan pola melalui tahap pelatihan dan penelitian (Anzihory, et al., 2016). Arsitektur JST memiliki 3 lapisanutama, yaitu lapisan masukan, lapisan tersembunyi dan laposan keluaran. Dalam prosesnya semua neuron pada *input layer* akan dihubungkan dengan semua *hidden layer*, kemudian *hidden layer* akan dihubungkan dengan semua *output layer*. *Hidden layer* akan dihubungkan dengan *activation layer* terlebih dahulu untuk menentukan objek pada *input layer* termasuk dalam kelas mana yang tersedia pada *output layer*. Jika dalam arsitektur JST memiliki lebih dari satu *hidden layer* maka dapat disebut *Multi Layer Perceptron* (MLP). Parameter pada JST ditentukan melalui *weight* dan *bias*, dimana *weight* merepresentasikan hubungan antar *input layer* dengan *hidden layer* dan *bias* merepresentasikan hubugan antar *hidden layer* dengan *output layer* (Sena, 2017). Arsitektur JST secara sederhana dapat dilihat pada Gambar 2.10.



Gambar 2.10 Arsitektur Jaringan Saraf Tiruan Sederhana

Sumber : (https://www.researchgate.net/)

Pada metode JST terdapat proses pembelajaran. Algoritma pembelajaran yang sering digunakan adalah algoritma *backpropagation.* Algoritma *backpropagation* merupakan salah satu algoritma pembelajaran terawasi yang biasanya digunakan pada model *Multi Layer Perceptron* untuk mendapatkan *weight* dan *bias*. Selain itu, algoritma *backpropagation* dikenal dengan tingkat pengenalan yang lebih akurat dibandingkan dengan algoritma pembelajaran lain, seperti *Learning Vector Quantization* (LVQ) (Fauzi, 2018). Proses pembelajaran algoritma *backpropagation* terdiri dari tiga fase, antara lain fase *feedforward, backpropagation,* dan *updating weight.* Sebelum melakukan proses pembelajaran, data yang dijadikan sumber pembelajaran algoritma *backpropagation* atau bisa disebut data latih akan dilakukan normalisasi terlebih dahulu. Normalisasi data dilakukan dengan tujuan agar fitur yang terdapat pada setiap data memiliki pengaruh yang sama rata atau dapat dikatakan tidak ada fitur yang lebih mendominasi diantara fitur lainnya (Fauzi, 2018). Persamaan 2.3 menunjukkan model normalisasi yang dipakai, yaitu *min-max normalization*.

$$D_{normal} = \frac{D - D_{min}}{D_{max} - D_{min}} \quad (2.3)$$

Keterangan:

$D_{normal} = nilai\ hasil\ normalisasi\ data$

$D = nilai\ data\ sebelum\ normalisasi$

$D_{min} = nilai\ terkecil\ dari\ seluruh\ data$

$D_{max} = nilai\ terbesar\ dari\ seluruh\ data$

Selain itu, jumlah neuron pada *hidden layer* juga harus ditentukan terlebih dahulu. Untuk melakukan hal tersebut dapat digunakan aturan yang dibuat oleh Heaton (2008), yang berbunyi:

1. Jumlah neuron harus berada diantara jumlah neuron pada *input layer* dan jumlah neuron pada *output layer*.
2. Jumlah neuron dapat ditentukan melalui 2/3 dari jumlah neuron pada *input layer* dan ditambah dengan jumlah neuron pada *output layer*.
3. Jumlah neuron harus berada dibawah dua kali jumlah neuron pada *input layer*.

Berdasarkan ketiga aturan tersebut, untuk menentukan jumlah neuron *hidden layer* dalam arsitektur Jaringan Saraf Tiruan pada penelitian ini dapat dituliskan pada Persaman 2.3.

$$\sum hN = round\left(\frac{2\sum iN}{3} + \sum oN\right) \quad (2.4)$$

Keterangan:

$\sum hN = Jumlah\ hidden\ Neuron$

$\sum iN = Jumlah\ input\ Neuron$

$$\sum oN = Jumlah\ output\ Neuron$$

Hal terakhir yang harus dilakukan sebelum memulai proses pembelajaran algoritma *backpropagation* adalah menentukan fungsi aktivasi. Penelitian ini akan menggunakan 2 fungsi aktivasi, yaitu fungsi aktivasi sigmoid digunakan pada *hidden layer*, sedangkan fungsi aktivasi softmax digunakan pada *output layer*. Fungsi aktivasi sigmoid dan turunannya ditunjukkan pada Persamaan 2.5 dan Persamaan 2.6.

$$sigmoid(x) = \frac{1}{1+e^{-x}} \quad (2.5)$$

$$dsigmoid(x) = sigmoid(x) \times (1 - sigmoid(x)) \quad (2.6)$$

Nilai *x* merepresentasikan nilai *input* persamaan fungsi aktivasi tersebut. Kedua fungsi aktivasi ini digunakan karena fungsi aktivasi sigmoid merupakan fungsi aktivasi yang biasanya digunakan dalam algoritma *backpropagation* (Fauzi, 2018). Selain itu, fungsi aktivasi softmax dapat membantu untuk melakukan klasifikasi lebih dari satu kelas. Fungsi aktivasi softmax ditunjukkan pada Persamaan 2.7.

$$softmax(x) = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}} \quad (2.7)$$

Setelah menentukan ketiga hal tersebut, proses pembelajaran algoritma *backpropagation* dapat dilakukan. Fase pertama yang dilakukan adalah *feedforward* yang ditunjukkan pada Persamaan 2.8, 2.9, 2.10 dan 2.11.

$$Z1 = W1 * X + B1 \quad (2.8)$$

$$A1 = sigmoid(Z1) \quad (2.9)$$

$$Z2 = W2 * A1 + B2 \quad (2.10)$$

$$A2 = softmax(Z2) \quad (2.11)$$

Persamaan 2.8 merepresentasikan perhitungan pada *input layer*, dimana X melambangkan nilai *input* jaringan, $W1$ melambangkan nilai *weight* dari *input layer*, $B1$ melambangkan nilai *bias* dari *input layer* dan $Z1$ melambangkan nilai *output* yang dihasilkan dari perhitungan tersebut. Kemudian Persamaan 2.9 merepresentasikan *output* dari hasil perhitungan fungsi aktivasi sigmoid yang dilambangkan dengan $A1$. Selanjutnya, nilai *output* tersebut akan digunakan pada Persamaan 2.10. Persamaan tersebut merupakan perhitungan pada *hidden layer*, dimana $W2$ melambangkan nilai *weight* dari *hidden layer*, $B2$ melambangkan nilai *bias* dari *hidden layer* dan $Z2$ melambangkan nilai *output* yang dihasilkan dari perhitungan tersebut. Hasil *output* perhitungan pada *hidden layer* akan digunakan pada Persamaan 2.11 melalui perhitungan fungsi aktivasi softmax dan mendapatkan nilai *output* jaringan.

Setelah mendapatkan nilai *output* jaringan, akan dilakukan perhitungan *error* untuk memperbaiki nilai *weight* dan *bias*. Untuk mendapatkan nilai *error*, dapat digunakan *loss function*. Salah satu jenis *loss function* yang biasa digunakan untuk

fungsi aktivasi sigmoid adalah *Cross Entropy* (CE). Perhitungan *cross entropy* ditunjukkan pada Persamaan 2.12.

$$CE = \sum_i y_i \log A2_i \quad (2.12)$$

Setelah melakukan perhitungan nilai *error*, proses selanjutnya akan masuk ke dalam fase *backpropagation* yang ditunjukkan pada Persamaan 2.13, Persamaan 2.14, Persamaan 2.15, Persamaan 2.16 dan Persamaan 2.17.

$$dCEs = A2 - y \quad (2.13)$$

$$dW2 = dCEs * A1 \quad (2.14)$$

$$dB2 = dCEs \quad (2.15)$$

$$dW1 = dCEs * W2 * dsigmoid(Z1) * X \quad (2.16)$$

$$dB1 = dCEs * dsigmoid(Z1) \quad (2.17)$$

Pada fase *backpropagation*, akan dilakukan perhitungan nilai turunan (*derivative*) terhadap nilai yang telah didapatkan pada fase *feedforward*. Persamaan 2.13 menjelaskan perhitungan nilai *derivative* untuk *cross entropy* dan fungsi aktivasi *softmax* yang dilambangkan dengan $dCEs$. Nilai tersebut didapatkan dengan mengurangkan nilai *output* jaringan yang dilambangkan dengan $A2$ dan nilai sebenarnya yang dilambangkan dengan y. Selanjutnya Persamaan 2.14 dan Persamaan 2.15 berturut-turut menjelaskan perhitungan nilai *derivative* untuk *weight* dan *bias* pada *hidden layer*. Sedangkan Persamaan 2.16 dan Persamaan 2.17 berturut-turut menjelaskan perhitungan nilai *derivative* untuk *weight* dan *bias* pada *input layer*.

Fase terakhir dalam proses pembelajaran algoritma *backpropagation* adalah melakukan *updating weight* yang ditunjukkan pada Persamaan 2.18, Persamaan 2.19, Persamaan 2.20 dan Persamaan 2.21.

$$nW1 = W1 - (lr * dW1) \quad (2.18)$$

$$nW2 = W2 - (lr * dW2) \quad (2.19)$$

$$nB1 = B1 - (lr * dB1) \quad (2.20)$$

$$nB2 = B2 - (lr * dB2) \quad (2.21)$$

Keterangan:

$nW1 = Nilai\ weight\ baru\ pada\ hidden\ layer$

$nW2 = Nilai\ weight\ baru\ pada\ output\ layer$

$nB1 = Nilai\ bias\ baru\ pada\ hidden\ layer$

$nB2 = Nilai\ bias\ baru\ pada\ output\ layer$

$lr = learning\ rate$

Proses *updating weight* dilakukan dengan menghitung selisih dari nilai *weight* maupun nilai *bias* dengan hasil perkalian antara nilai *learning rate* dan nilai *derivative* dari *weight* dan *bias*. Nilai *learning rate* merupakan tingkat kedetailan

pembelajaran yang dilakukan oleh algoritma *backpropagation*. Nilai tersebut bisa didapatkan melalui referensi maupun percobaan eksperimen dengan mengambil nilai *learning rate* yang menghasilkan akurasi pembelajaran tertinggi.

Apabila proses *updating weight* pada pembelajaran algoritma *backpropagation* telah selesai, maka nilai *weight* dan *bias* yang terdapat pada *hidden layer* maupun *output layer* dapat ditemukan. Proses yang telah dipaparkan merupakan proses dalam satu siklus pelatihan, atau dapat disebut *epoch*. Proses tersebut akan diulang terus menerus sampai nilai *error* yang dihasilkan telah sesuai dengan yang diinginkan.

BAB 3 METODOLOGI

3.1 Tipe Penelitian

Menurut penjelasan pada Bab pendahuluan, tipe penelitian yang digunakan adalah tipe penelitian implementatif pengembangan lanjut. Yang dimaksud implementatif disini adalah penelitian yang mengimplementasikan suatu perangkat lunak maupun perangkat keras terhadap sistem yang akan dibangun. Dalam penelitian ini implementatif yang dimaksud adalah membangun sistem yang dapat mendetieksi penyakit aritmia jenis fibrilasi atrium secara otomatis. Bagian pengembangan lanjut dalam penelitian implementatif sendiri merupakan kegiatan penelitian untuk membuat produk yang dikembangkan dari sistem yang sudah ada. Penelitian ini mengacu pada saran dari penelitian sebelumnya yang dilakukan oleh Sofiana *et. al* (2020) yaitu dengan melakukan deteksi pada penyakit yang sama dengan menggunakan fitur dan metode klasifikasi yang berbeda. Pada peneltian tersebut fitur yang digunakan adalah berdasarkan nilai BPM, serta nilai rata-rata dan nilai median dari interval RR. Kemudian sebanyak 24 data latih digunakan untuk menghasilkan *hyperplane* yang digunakan untuk metode klasifikasi SVM (*Support Vector Machine*). Sementara hasil yang didapatkan pada penelitian tersebut dari 12 data yang diujikan adalah sistem dapat melakukan klasifikasi dengan tingkat akurasi sebesar 83,33% dengan waktu rata-rata yang dibutuhkan pada tahap training adalah 219,30 ms dan pada tahap testing adalah 0,09 ms.

Oleh karena itu, pengembangan lanjut yang akan dilakukan adalah dengan menggunakan fitur dan metode selain penelitian sebelumnya. Fitur statistik digunakan untuk mendapatkan nilai interval dan gradien QRS. Kemudian dari nilai interval dan gradient QRS akan dihitung masing-masing nilai mean dan median dari 6 siklus yang terdeteksi. Sedangkan metode yang digunakan untuk klasifikasi penyakit Fibrilasi Atrium adalah metode Jaringan Saraf Tiruan.

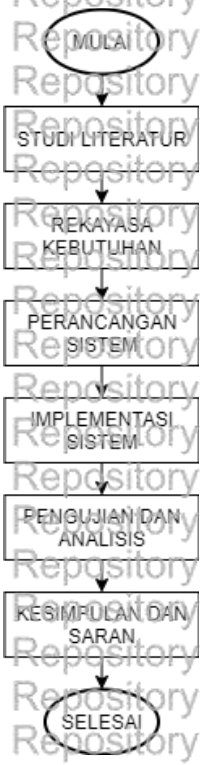
3.2 Strategi dan Rancangan Penelitian

Dalam melakukan penelitian ini, diperlukan langkah struktural yang dirancang dengan memperhatikan kebutuhan penelitian. Langkah struktural yang dirancang meliputi segala kebutuhan yang berkaitan dengan penelitian ini. Pada prosesnya, perancangan tersebut dilakukan dengan percobaan yang menyesuaikan tujuan dari sistem yang dibangun.

3.2.1 Metode Umum

Dalam penelitian ini memiliki tahapan-tahapan yang dilakukan untuk kebutuhan penelitian. Tahapan tersebut dilakukan agar penelitian yang dilakukan memiliki pondasi yang kuat serta memiliki arah yang jelas. Pada tahap pertama pertama penelitian yang dilakukan adalah mencari studi literatur sebagai landasan kepustakaan yang berguna sebagai literatur pendukung penelitian. Kedua, menganalisis kebutuhan sistem secara keseluruhan, baik kebutuhan fungsional

maupun kebutuhan non-fungsional. Ketiga, merancangan sistem sesuai dengan kebutuhan yang telah dianalisis. Keempat, mengimplementasikan sistem sesuai dengan sistem yang dirancang pada proses sebelumnya. Kelima, melakukan pengujian untuk mengetahui kinerja sistem yang telah diimplementasikan dan menganalisis data tersebut apakah sudah memenuhi tujuan dibuatnya sistem. Kesimpulan akan didapatkan setelah melakukan pengujian serta akan muncul saran apabila menjadi acuan bagi penelitian selanjutnya yang menutup tahapan pada penelitian ini. Gambar 3.1 menunjukkan tahapan penelitian yang dilakukan.



Gambar 3.1 *Flowchart* penelitian

3.2.2 Subjek Penelitian

Penelitian yang dilakukan membutuhkan subjek yang terbagi menjadi dua kondisi, yaitu orang-orang yang menderita penyakit aritmia jenis Fibrilasi Atrium (FA) serta orang-orang dengan kondisi normal. Jenis penyakit tersebut terjadi saat atrium jantung berdetak lebih cepat dan detak jantungnya tidak beraturan.

3.2.3 Lokasi Penelitian

Lokasi penelitian untuk pengambilan data yang nantinya akan dibagi menjadi data latih dan data uji pada penelitian ini adalah dengan mengakuisisi sinyal EKG normal melalui sensor AD8232 yang dilakukan di kediaman subjek.

3.2.4 Teknik Pengumpulan Data

Dataset yang dikumpulkan terdiri dari kondisi normal dan kondisi fibrilasi atrium. Oleh karena itu, hasil sinyal EKG kondisi normal akan diakusisi langsung dengan sensor AD8232 pada subjek dengan kondisi normal, sedangkan hasil sinyal

EKG penderita FA akan diambil melalui database *online* yaitu *physionet.org/content/afdb/1.0.0/* dan *physionet.org/content/ltafdb/1.0.0/* yang kemudian akan dibandingkan melalui fitur yang telah ditentukan pada penelitian ini. Karena penderita FA dengan orang normal memiliki bentuk sinyal EKG yang berbeda.

3.2.5 Teknik Analisis Data dan Pembahasan

Teknik analisis yang dilakukan pada penelitian ini adalah melalui perhitungan rata-rata (*Mean*) dan nilai tengah (*Median*) dari dua parameter yang digunakan, yaitu Interval QRS dan Gradien QRS. Kedua parameter tersebut akan disimpan pada *buffer* untuk mendapatkan nilai *mean* dan *median*nya yang kemudian akan digunakan sebagai masukan pada metode klasifikasi Jaringan Saraf Tiruan.

3.2.6 Peralatan Pendukung yang Digunakan

Untuk melakukan sebuah penelitian sistem yang dibangun tentunya membutuhkan peralatan pendukung dalam merancang sistem dan menerapkan rancangan tersebut. Terdapat beberapa peralatan pendukung yang digunakan antara lain:

3.2.6.1 Perangkat Keras

1. PC (*Personal Computer*) atau Laptop
2. Sensor AD8232
3. Elektrode
4. Mikrokontroler Arduino Uno
5. LCD 16x2 *with* I2C
6. Kabel *Jumper*
7. *Breadboard*

3.2.6.2 Perangkat Lunak

1. Arduino IDE
2. Pycharm IDE
3. Library LCD 16x2
4. Library I2C

BAB 4 REKAYASA KEBUTUHAN

Pada bab rekayasa kebutuhan akan dijelaskan mengenai gambaran umum dari sistem yang akan dibangun, rekayasa kebutuhan sistem baik secara fungsional maupun non-fungsional dan batasan dari desain sistem yang akan dibangun.

4.1 Gambaran Umum Sistem

Sistem pendeteksi Fibrilasi Atrium merupakan sistem yang dapat mendeteksi apakah pengguna dari sistem memiliki detak jantung dengan kondisi fibrilasi atrium atau tidak. Prinsip kerja dari sistem untuk mendeteksi penyakit fibrilasi atrium adalah dengan cara mengolah hasil sinyal EKG yang terbaca oleh sensor AD8232. Sinyal EKG yang dihasilkan oleh sensor AD8232 selanjutnya akan dianalisis untuk mendapatkan nilai dari 4 fitur yang telah ditentukan, yaitu Rata-rata dan Median dari Interval QRS, serta Rata-Rata dan Median dari Gradien QRS. Nilai rata-rata dan median didapatkan dari 6 siklus yang disimpan ke dalam *buffer* oleh sistem dan kemudian akan dilakukan perhitungan untuk mendapatkan nilai mean dan mediannya. Selanjutnya apabila nilai dari keempat fitur tersebut telah didapatkan maka selanjutnya akan diproses menggunakan metode Jaringan Saraf Tiruan. Proses yang dihasilkan pada metode Jaringan Saraf Tiruan terbagi atas 2 kelas, yaitu kondisi normal dan kondisi fibrilasi atrium.

Kondisi tersebut akan ditampilkan pada LCD 16x2 yang terdapat pada sistem untuk memberikan informasi kepada pengguna sistem sekaligus menandakan bahwa hasil sinyal EKG yang terbaca oleh sensor AD8232 telah selesai diolah. Oleh karena itu, terdapat empat poin penting yang harus dapat dilakukan sistem, antara lain membaca sinyal EKG pada subjek penelitian, mengolah hasil sinyal EKG tersebut untuk mendapatkan nilai dari 4 fitur yang digunakan, melakukan klasifikasi terhadap hasil sinyal EKG tersebut untuk menentukan subjek penelitian memiliki kondisi jantung normal ataupun kondisi fibrilasi atrium dan menampilkan hasil klasifikasi pada LCD 16x2. Gambar 4.1 menunjukkan diagram blok dari sistem pendeteksi Fibrilasi Atrium yang dibangun.

Gambar 4.1 Blok Diagram Sistem

4.2 Analisis Kebutuhan

Analisis kebutuhan dilakukan dengan tujuan mengetahui segala hal yang dibutuhkan untuk merancang sistem pendeteksi Fibrilasi Atrium. Analisis

kebutuhan pada sistem yang akan dibangun terbagi atas kebutuhan fungsional dan kebutuhan non-fungsional.

4.2.1 Kebutuhan Fungsional

Kebutuhan fungsional dari sistem pendeteksi Fibrilasi Atrium terdiri atas tujuan dan fungsi yang diharapkan dari sistem ini, diantaranya:

1. Sistem mampu mengakuisisi sinyal EKG pada jantung.

   Sistem yang dirancang akan digunakan untuk mengklasifikasikan kondisi jantung normal dan kondisi jantung FA. Dalam prosesnya dibutuhkan sinyal EKG sebagai input yang selanjutnya akan dilakukan analisis untuk mendapatkan nilai fitur pada sinyal EKG tersebut. Oleh karena itu sistem harus dapat mengakuisisi sinyal EKG pada subjek untuk dapat melanjutkan proses selanjutnya yaitu ekstraksi fitur.

2. Sistem mampu mencari nilai fitur mean interval QRS.

   Setelah sistem mampu mengakuisisi sinyal EKG, maka proses selanjutnya sistem yang dirancang harus mampu mendapatkan nilai fitur yang digunakan, salah satunya adalah mean interval QRS. Untuk mendapatkan nilai mean interval QRS sistem akan menyediakan buffer untuk menampung nilai interval QRS sebanyak 6 siklus. Kemudian akan dilakukan perhitungan untuk mendapatkan nilai mean atau nilai rata-rata dari nilai interval QRS yang tertampung pada buffer tersebut.

3. Sistem mampu mencari nilai fitur median interval QRS.

   Pada saat yang sama sistem juga harus mendapatkan nilai median interval QRS. Dari nilai interval QRS yang didapatkan sistem juga akan menyediakan buffer untuk menampung hasil pengurutan dari 6 siklus nilai interval QRS yang dideteksi. Kemudian sistem akan melakukan perhitungan dengan menambahkan indeks ke-2 dan ke-3 dan membagi 2 nilai pada indeks tersebut untuk mendapatkan nilai median dari interval QRS yang tertampung pada buffer tersebut.

4. Sistem mampu mencari nilai fitur mean gradien QRS.

   Untuk fitur selanjutnya yaitu mean gradien QRS, sistem yang dirancang juga menyediakan buffer untuk menampung nilai gradien QRS sebanyak 6 siklus yang telah berhasil didapatkan. Setelah itu sistem akan melakukan perhitungan untuk mendapatkan nilai mean dari gradien QRS apabila buffer yang disediakan sudah penuh.

5. Sistem mampu mencari nilai fitur median gradien QRS.

   Untuk mendapatkan nilai fitur median gradien QRS, sistem yang dirancang akan menyediakan buffer untuk menyimpan nilai yang telah diurutkan dari gradien QRS yang dideteksi. Kemudian dari data yang telah diurutkan dan sistem harus mampu melakukan perhitungan yang sama seperti saat sistem mendapatkan nilai median interval QRS.

6. Sistem dapat mengimplementasikan metode Jaringan Saraf Tiruan

   Setelah berhasil mendapatkan nilai dari ekstraksi fitur yang digunakan, sistem yang dirancang harus dapat mengolah nilai tersebut untuk melakukan pengklasifikasian kondisi jantung normal ataupun kondisi jantung FA. Melalui pelatihan data dengan algoritma *backpropagation* sistem akan mendapatkan model dari arsitektur JST dengan nilai *weight* dan *bias* yang tetap. Kemudian hasil yang diinginkan adalah sistem mampu melakukan klasifikasi terhadap dua kelas, yaitu kelas "FA" dan kelas "Normal".

7. Sistem dapat menampilkan hasil klasifikasi melalui LCD 16x2.

   Setelah melakukan klasifikasi, sistem harus dapat menampilkan hasil klasifikasi yang dilakukan beserta waktu komputasi yang dibutuhkan dalam melakukan perhitungan untuk menentukan klasifikasi tersebut. Hasil klasifikasi dan waktu komputasi akan ditampilkan melalui LCD 16x2.

4.2.2 Kebutuhan Non-Fungsional

Kebutuhan Non-Fungsional merupakan kebutuhan dari sistem yang berkaitan dengan perangkat keras maupun perangkat lunak yang dapat membantu sistem mencapai tujuan yang diharapkan. Kebutuhan perangkat keras dari sistem deteksi Fibrilasi Atrium antara lain:

1. Laptop atau *Personal Computer* (PC)

   Laptop atau *Personal Computer* (PC) dibutuhkan sebagai media yang membantu dalam membuat program untuk sistem pendeteksi Fibrilasi Atrium. Selain itu Laptop atau *Personal Computer* (PC) juga dibutuhkan untuk memberikan daya pada sistem sehingga sistem dapat berfungsi dengan semestinya. Adapun spesifikasi Laptop yang digunakan pada penelitian ini dapat dilihat pada Tabel 4.1.

Tabel 4.1 Spesifikasi Laptop untuk Penelitian

Spesifikasi	Keterangan
Merk	Lenovo G410
CPU	Intel i5-4200M
Memory	6 GB
Storage	500 GB
Operating System	Windows 10

2. Mikrokontroler Arduino Uno

   Mikrokontroler Arduino Uno dibutuhkan sebagai salah satu komponen utama yang berfungsi untuk mengolah data hasil sinyal EKG yang diterima dari sensor AD8232. Digunakannya Arduino Uno sebagai mikrokontroller pada sistem ini karena Arduino Uno dapat menampung kebutuhan Pin baik

dari sensor AD8232 maupun dari LCD 16x2. Selain itu kapasitas memory pada Arduino Uno dirasa sudah cukup untuk menampung kode program untuk melakukan komputasi pada sinyal EKG yang dideteksi oleh sensor AD8232 serta terdapat *ADC Converter* untuk menerima sinyal analog dari sensor AD8232 namun tetap melakukan komputasi secara digital. Tabel 4.2 menunjukkan Spesifikasi dari Arduino Uno.

Tabel 4.2 Spesifikasi Arduino Uno R3

Spesifikasi	Keterangan
B Operasi	5V
Tegangan Masukan (Rekomendasi)	7-12V
Tegangan Masukan (Batas)	6-20V
Pin Digital I/O	14 (disediakan 6 pin PWM output)
Pin Digital Masukan	6
Arus DC tiap Pin I/O	40 mA
Arus DC untuk Pin 3.3V	50 mA
Flash Memory	32KB
Static RAM	2 KB
EEPROM	1 KB
Kecepatan clock	16 MHz

3. Sensor AD8232

Sensor AD8232 dibutuhkan sebagai komponen yang mendeteksi sinyal EKG jantung manusia. Sensor AD8232 memiliki tiga buah elektrode untuk mendeteksi aktivitas listrik jantung pada manusia. Digunakannya sensor AD8232 pada sistem ini karena pada penelitian sebelumnya terkait pendeteksian penyakit jantung, sensor ini merupakan yang paling sering digunakan. Selain itu pada sensor ini sudah terdapat filter sehingga memiliki tingkat akurasi yang baik dalam mendeteksi sinyal EKG pada jantung. Pin yang terdapat pada sensor ini juga dapat mengirimkan secara langsung data yang dideteksi melalui Pin "output" nya. Tabel 4.3 menunjukkan spesifikasi sensor AD8232.

Tabel 4.3 Spesifikasi Sensor AD8232

Spesifikasi	Keterangan
Kisaran Tegangan Suplai	2-3.5V
Kisaran Suhu yang Ditentukan	0 - 70°C
Kisaran Suhu Operasional	-40 – 85°C

4. LCD 16x2 *with* I2C

LCD 16x2 *with* I2C dibutuhkan sebagai komponen yang akan menampilkan *output* hasil klasifikasi yang dilakukan oleh Mikrokontroler Arduino Uno. Penggunaan LCD 16x2 with I2C daripada LCD 16x2 biasa bertujuan untuk mengurangi pemakaian pin pada Arduino Uno sehingga dengan otomatis mengurangi pemakaian kabel jumper yang berlebihan.

5. *Connector* Mikrokontroler dengan USB Laptop

Connector Mikrokontroler dengan USB Laptop dibutuhkan sebagai media perantara dalam mengupload program pada Arduino IDE ke dalam Mikrokontroler. Selain itu komponen tersebut juga dibutuhkan sebagai perantara *power supply* untuk sistem agar dapat bekerja sesuai dengan yang diharapkan.

Sementara untuk kebutuhan perangkat lunak dari sistem deteksi Fibrilasi Atrium antara lain:

1. Sistem Operasi *Windows 10*

Sistem Operasi *Windows 10* dibutuhkan sebagai pengoperasian IDE yang digunakan untuk membuat program utama pada sistem pendeteksi Fibrilasi Atrium. Selain itu Sistem Operasi *Windows 10* juga lebih kompatibel dengan aplikasi yang digunakan untuk membuat desain *packaging* untuk sistem pendeteksi Fibrilasi Atrium.

2. Arduino IDE

Arduino IDE merupakan perangkat lunak yang akan dijalankan pada Sistem Operasi *Windows 10* dan berfungsi sebagai IDE yang kompatibel dengan Mikrokontroler sehingga program akan ditulis dan diupload melalui perangkat lunak ini.

3. Pycharm IDE

Pycharm IDE merupakan perangkat lunak yang akan digunakan pada tahap pelatihan data latih metode Jaringan Saraf Tiruan. Bahasa pemrograman yang digunakan pada IDE ini adalah bahasa pemrograman python 3.9.

4.3 Batasan Desain Sistem

Batasan desain sistem merupakan batasan maksimal yang dapat dijangkau oleh sistem ini yang dilihat dari aspek fungsional maupun non-fungsionalnya. Batasan desain pada Sistem Pendeteksi Fibrilasi Atrium meliputi:

1. Sistem menggunakan sebanyak 3 buah elektrode dalam mendeteksi aktivitas listrik jantung pada subjek penelitian.
2. Sistem hanya dapat mendeteksi ketika elektrode telah terpasang pada subjek penelitian.

3. Subjek yang akan dideteksi disarankan untuk berada dalam kondisi diam agar pendeteksian dapat dilakukan dengan maksimal.
4. Subjek yang dideteksi disarankan untuk tidak mengunakan atau memakai barang yang terbuat dari logam agar tidak ada *noise* saat pendeteksian sinyal jantung.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

5.1 Perancangan Sistem

Pada proses pembuatan sistem pendeteksi Fibrilasi Atrium terdapat tahapan perancangan dan implementasi yang akan dijelaskan pada Bab 5. Proses perancangan dan implementasi pada sistem pendeteksi Fibrilasi Atrium tidak hanya dari perangkat lunaknya saja melainkan juga dari perangkat keras nya.

5.1.1 Perancangan *Packaging* Sistem

Pada perancangan *packaging* sistem dari sistem pendeteksi Fibrilasi Atrium dapat dilihat pada Gambar 5.1.

Gambar 5.1 *Packaging* Sistem

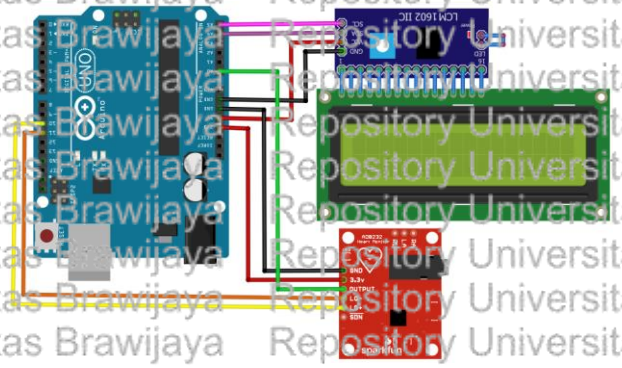
Sistem pendeteksi Fibrilasi Atrium akan dikemas dalam *packaging* sesuai pada Gambar 5.1 dimana *packaging* tersebut memiliki bahan dasar akrilik. Bahan dasar tersebut digunakan karena memiliki daya tahan yang cukup kuat, bobotnya yang ringan serta memiliki harga yang terjangkau. *Packaging* berbentuk balok dengan ukuran 14 cm x 10 cm x 8,5 cm. Selain itu, *packaging* memiliki 4 buah lubang. 1 lubang digunakan untuk LCD 16x2 sebagai informasi status sistem, 1 lubang digunakan untuk *connector* Arduino dengan USB Laptop, 1 lubang lagi digunakan untuk kabel elektrode yang dihubungkan antara sensor AD8232 dengan subjek penelitian dan 1 lubang tambahan apabila ingin menggunakan adaptor sebagai *power supply* eksternal. Elemen *packaging* pada tiap sisinya terhubung dengan lem G, sementara perangkat keras akan diletakkan di dalamnya dengan bantuan *double tape*.

5.1.2 Perancangan Perangkat Keras Sistem

Proses perancangan perangkat keras pada sistem pendeteksi fibrilasi atrium terbagi atas dua perancangan, yaitu perancangan untuk mikrokontroler dan perancangan untuk pemasangan elektrode pada subjek penelitian. Nantinya setiap elemen perangkat keras akan terhubung melalui kabel *jumper*, sementara elektrode akan terhubung dengan sensor AD8232 melalui 3 buah kabel khusus yang akan dimasukkan ke dalam lubang pada sensor AD8232.

5.1.2.1 Perancangan Perangkat Keras Mikrokontroler

Gambar 5.2 menunjukkan rancangan dari perangkat keras yang membangun sistem deteksi Fibrilasi Atrium.



Gambar 5.2 Skematik Perangkat Keras Mikrokontroler

Pada Gambar 5.2 dapat dilihat rangkaian skematik dari sistem deteksi fibrilasi atrium yang terdiri dari Mikrokontroler Arduino Uno R3, Sensor AD8232 dan LCD 16x2 *with* I2C. Setiap komponen dihubungkan oleh kabel jumper dengan ketentuan pin yang disesuaikan spesifikasi Arduino Uno, sensor AD8232 dan LCD 16x2 *with* I2C sehingga sistem dapat berjalan sesuai dengan yang diharapkan. Kabel jumper yang digunakan adalah kombinasi kabel jumper *female male*. Tabel 5.1 menunjukkan hubungan setiap Pin antara sensor AD8232 dan Arduino Uno.

Tabel 5.1 Keterangan Pin Sensor AD8232 dengan Arduino Uno

Pin Arduino Uno R3	Pin Sensor AD8232
3.3V	3.3V
GND	GND
A0	OUTPUT
LO+	10
LO-	11

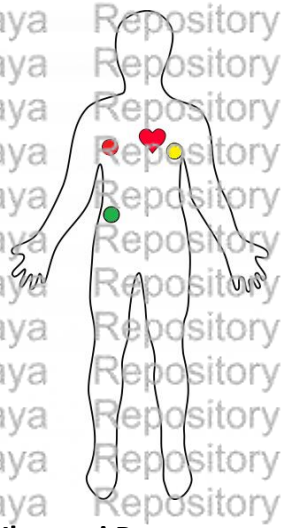
Sedangkan ketentuan pin Arduino Uno dengan LCD 16x2 akan dibantu dengan I2C untuk menghemat penggunaan pin pada Arduino Uno sehingga pada pengoperasiannya I2C yang akan memberikan instruksi kepada LCD 16x2. Tabel 5.2 menunjukkan hubungan setiap Pin antara LCD 16x2 dengan Arduino Uno.

Tabel 5.2 Keterangan Pin LCD 16x2 *with* I2C dengan Arduino Uno

Pin Arduino Uno R3	Pin Modul I2C
5V	5V
GND	GND
A4	SDA
A5	SCL

5.1.2.2 Perancangan Penempatan Elektrode

Perancangan penempatan elektrode pada sistem pendeteksi Fibrilasi Atrium bertujuan agar pendeteksian aktivitas listrik pada jantung yang dilakukan oleh sensor AD8232 dapat bekerja secara maksimal. Penempatan elektrode pada sistem pendeteksi Fibrilasi Atrium menyesuaikan dengan jumlah elektrode yang digunakan yaitu 3 buah elektrode. Penempatan elektrode akan ditempatkan pada bagian tubuh subjek yang dapat dilihat pada ilustrasi sesuai gambar 5.3.



Gambar 5.3 Ilustrasi Penempatan Elektrode

Sumber : (https://how2electronics.com/)

Pada Gambar 5.3 terdapat 3 buah elektrode yang memiliki warna yang berbeda untuk setiap elektrodenya. Perbedaan warna tersebut bertujuan untuk membedakan penempatan masing-masing elektrode. Hubungan perbedaan warna dengan penempatan pada bagian tubuh subjek ditunjukkan pada Tabel 5.3.

Tabel 5.3 Penempatan Elektrode Sesuai Warna

Posisi	Warna Elektrode
Dada Kanan Atas	Merah
Dada Kiri Atas	Kuning
Perut Kanan Bawah	Hijau

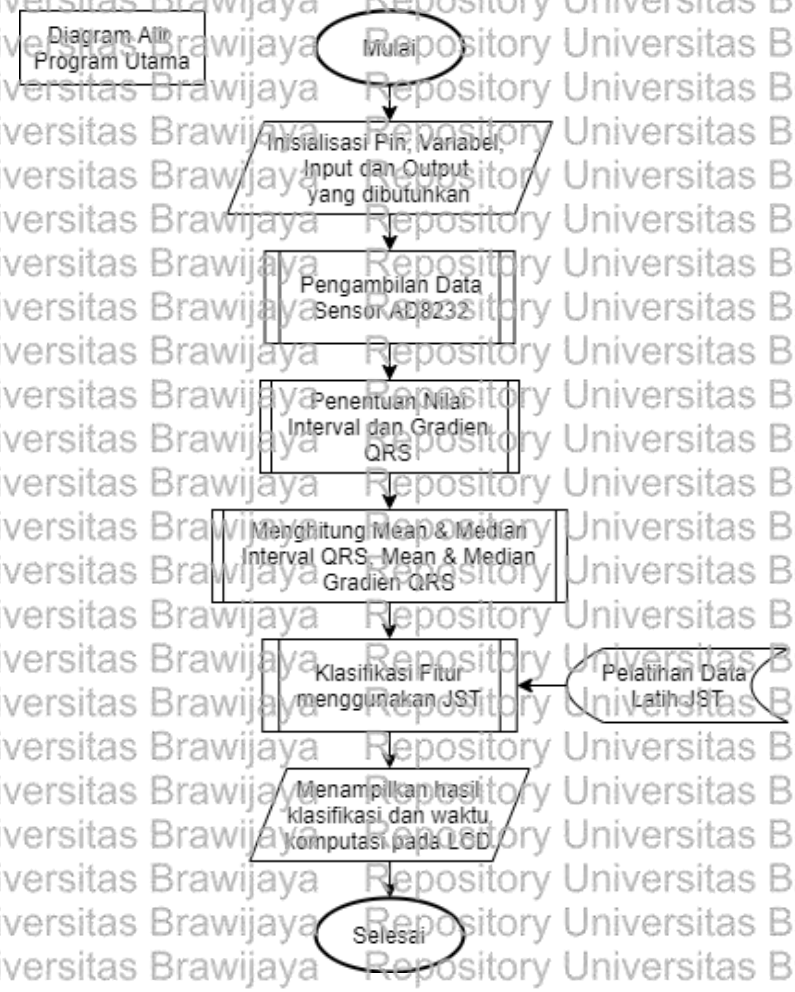
5.1.3 Perancangan Perangkat Lunak Sistem

Pada perancangan perangkat lunak sistem akan dijelaskan mengenai diagram alir program dari sistem pendeteksi Fibrilasi Atrium. Perancangan perangkat lunak pada sistem pendeteksi Fibrilasi Atrium sendiri memiliki program utama yang terdiri dari beberapa program pendukung.

5.1.3.1 Perancangan Program Utama

Pada bagian program utama, pin, variable, masukan dan keluaran akan diinisialisasi. Hal tersebut dilakukan untuk menentukan pin mana yang akan

menjadi jembatan untuk pertukaran data yang terjadi pada perangkat keras dan variabel mana yang akan menyimpan data tersebut. Sedangkan untuk data latih akan digunakan untuk pelatihan data untuk metode Jaringan Saraf Tiruan. Setelah itu sistem akan mengambil data melalui sensor AD8232 dan mengirimkan data tersebut ke mikrokontroller Arduino. Kemudian pada proses ekstraksi fitur mikrokontroler Arduino akan melakukan komputasi untuk mendapatkan keempat nilai fitur yang ditentukan, yaitu Rata-rata dan Median dari Interval QRS, serta Rata-rata dan Median dari Gradien QRS. Gambar 5.4 menunjukkan diagram alir dari program utama



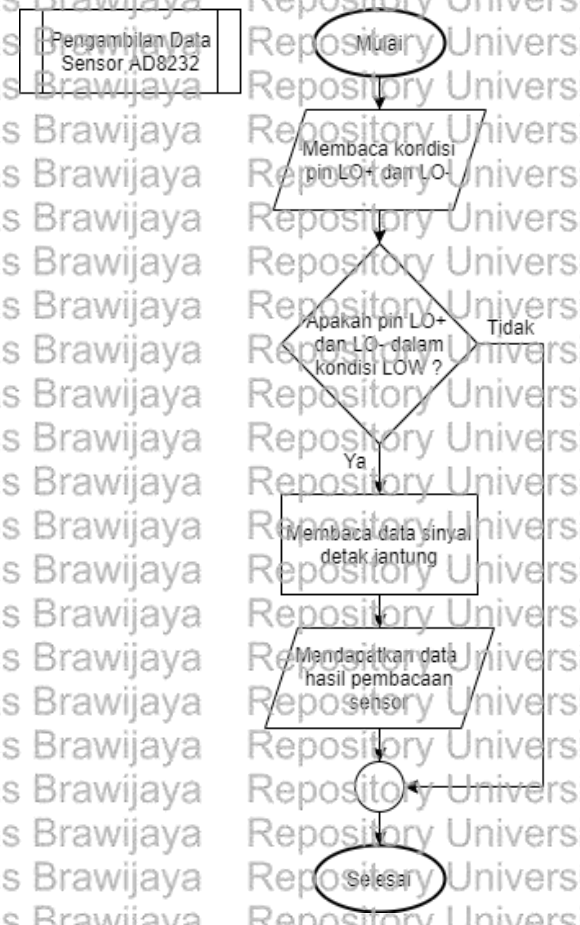
Gambar 5.4 Diagram Alir Program Utama

Setelah mendapatkan nilai fitur yang ditentukan, selanjutnya sistem akan melakukan klasifikasi dengan metode JST, dimana akan dilakukan komputasi apakah nilai fitur yang menjadi masukan termasuk ke dalam kelas Normal ataupun kelas Fibrilasi Atrium. Pada saat melakukan klasifikasi, sistem akan membandingkan nilai fitur yang baru masuk dengan data latih yang telah dilakukan pelatihan sebelumnya. Proses pelatihan dilakukan dengan Bahasa pemrograman python. Sedangkan kode program lainnya dilakukan dengan Bahasa pemrograman C. Setelah dilakukan klasifikasi maka LCD 16x2 akan menampilkan hasil beserta waktu komputasinya.

5.1.3.2 Perancangan Pengambilan Data melalui Sensor AD8232

Pada perancangan pengambilan data melalui sensor AD8232 mula-mula dilakukan inisialisasi tiga variabel yang masing-masing berfungsi untuk pengecekan pin dan menyimpan nilai yang dari sensor AD8232. Pin tersebut pada AD8232 berfungsi untuk menghasilkan sinyal dari jantung yang dideteksi melalui elektrode plus (untuk pin LO+) dan elektrode minus (untuk pin LO-). Sinyal yang dihasilkan dari kedua pin tersebut adalah sinyal LOW, sehingga akan dilakukan pengecekan apabila kedua pin bernilai LOW maka sensor akan membaca sinyal jantung dari subjek dan mengirimkannya ke arduino melalui pin OUTPUT. Gambar 5.5 menunjukkan diagram alir sensor AD8232 dalam mengambil data subjek.

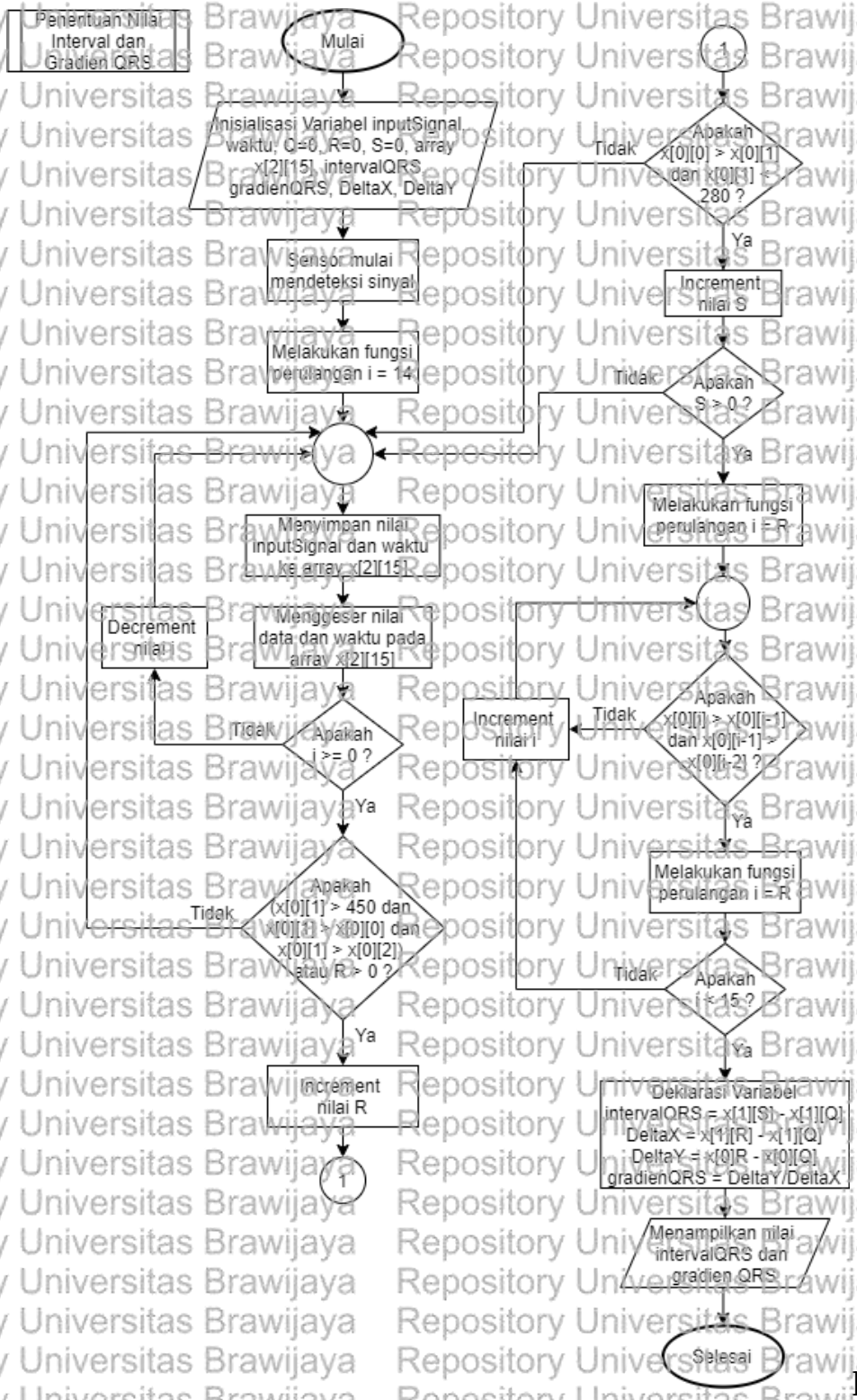
Sinyal EKG yang diakuisisi oleh sensor AD8232 nantinya dapat dilihat pada *serial plotter* melalui laptop. Sinyal EKG yang diakuisisi tersebut tidak menggunakan filter tambahan dari perangkat lunak karena pada sensor AD8232 sudah terdapat rangkaian filter berupa *low pass* dan *high pass* filter. Oleh karena itu, sinyal EKG pada *serial plotter* akan terlihat sedikit *noise* saja.



Gambar 5.5 Diagram Alir Pengambilan Data melalui Sensor AD8232

5.1.3.3 Perancangan Ekstraksi Fitur Interval dan Gradien QRS

Gambar 5.6 menunjukkan ektraksi fitur yang dirancang untuk mendapatkan interval dan gradien QRS.



Gambar 5.6 Diagram Alir Ekstraksi Fitur Interval dan Gradien QRS

Pada perancangan ekstraksi fitur interval dan gradien QRS, sistem akan mendapatkan nilai interval QRS dan gradien QRS dari sinyal yang telah berhasil dideteksi pada subprogram sebelumnya. Langkah awal ntuk mendapatkan nilai interval dan gradien QRS adalah menginisialisasi variabel yang dibutuhkan.

Variabel tersebut antara lain inputSignal, Q, R, S, array x[2][15], DeltaX, DeltaY, intervalQRS dan gradienQRS. Setelah itu sensor akan mendeteksi sinyal listrik pada jantung.

Dari hasil deteksi yang dilakukan oleh sensor, sistem akan mencari titik Q, R dan S terlebih dahulu sebelum menentukan interval dan gradiennya. Untuk itu, data hasil deteksi sensor akan dimasukkan ke dalam variabel inputSignal dan fungsi millis dimasukkan ke dalam variabel waktu. Hal tersebut dilakukan untuk menyimpan nilai amplitude beserta pewaktuannya. Setelah data dan fungsi millis berhasil disimpan ke dalam variabel inputSignal dan waktu, maka akan dilakukan proses perulangan. Pada proses perulangan, nilai dari amplitudo beserta waktunya akan dimasukkan ke dalam array x[2][15]. Nilai dari variabel inputSignal akan dimasukkan ke dalam array x[0][i], sementara nilai dari variabel waktu akan dimasukkan ke dalam array x[1][i]. Proses perulangan sendiri menggunakan fungsi *decrement*, dimana nilai i dimulai dari 14 dan akan melakukan *decrement* nilai i di setiap perulangan hingga i bernilai 0.

Setelah nilai amplitudo dan waktu disimpan pada array x[2][15], program akan mulai mencari nilai dari titik R dengan seleksi kondisi perulangan. Di dalam seleksi kondisi perulangan, akan diperiksa apakah nilai pada array x[0][1] lebih besar dari 420, lebih besar dari nilai sebelumnya dan nilai setelahnya. Selain itu, nilai dari variabel R juga akan diperiksa apakah bernilai lebih besar dari 0. Apabila kondisi tidak terpenuhi, maka nilai hasil deteksi beserta fungsi millis yang disimpan pada variabel waktu akan disimpan kembali. Namun apabila kondisi terpenuhi, maka akan dilakukan *increment* terhadap nilai dari variabel R. Pada proses pencarian titik R, digunakan nilai *threshold* sebesar 420. Nilai tersebut didapatkan dari hasil analisis terhadap hasil deteksi yang dilakukan oleh sensor. Sementara kondisi perulangan ditentukan dengan posisi titik R yang merupakan titik tertinggi dari satu siklus PQRST. Sehingga nilai pada titik R lebih tinggi dari nilai sebelumnya dan juga nilai setelahnya. Selain itu, nilai pada titik R juga harus lebih tinggi dari titik P dan titik T. Pada Gambar 5.7 yang merepresentasikan hasil deteksi menggunakan sensor, dapat dilihat titik R selalu berada diatas 420 yang ditandai dengan garis berwarna merah. Sementara untuk titik P dan titik T berada di kisaran 300 sampai dengan 350.

Setelah melakukan *increment* terhadap nilai dari variabel R, terdapat seleksi kondisi kembali untuk mendapatkan titik S. Di dalam seleksi kondisi akan diperiksa apakah nilai pada array x[0][0] lebih besar dari array x[0][1] dan nilai dari array x[0][-1] lebih kecil dari 280. Jika kondisi terpenuhi, maka akan dilakukan *increment* terhadap nilai dari variabel S. Namun apabila kondisi tidak terpenuhi maka akan dilakukan *increment* terhadap nilai dari variabel R. Pada Gambar 5.7 digunakan nilai *threshold* sebesar 280 untuk pencarian titik S yang ditandai warna hijau. Nilai tersebut didapatkan dari hasil analisis sinyal EKG dimana titik S selalu bernilai lebih kecil dari titik sebelumnya dan titik setelahnya. Gambar 5.7 juga menunjukkan bahwa nilai titik S berada di kisaran 240 sampai dengan kisaran 280.

Gambar 5.7 Sinyal EKG Hasil Deteksi Sensor

Setelah melakukan *increment* terhadap nilai dari variabel S dan mendapatkan nilai dari titik S, maka selanjutnya mencari titik Q. Pada proses pencarian titik Q, dilakukan seleksi kondisi untuk memeriksa apakah nilai dari variabel S lebih dari 0. Apabila kondisi tersebut tidak terpenuhi, maka akan kembali ke langkah saat memasukkan nilai variabel inputSignal dan waktu ke dalam array x[2][15]. Namun jika terjadi sebaliknya, maka akan dilakukan perulangan untuk mencari titik Q.

Pada kondisi perulangan, nilai variabel R yang telah didapatkan akan disimpan pada variabel i dan apabila i bernilai lebih dari 15 maka akan dilakukan *increment* terhadap nilai I sampai I bernilai 14. Pada perulangan tersebut, terdapat seleksi kondisi untuk menentukan nilai dari titik Q. Apabila nilai dari array x[0][i] lebih besar dari 2 nilai indeks sebelumnya maka 1 nilai indeks sebelumnya akan disimpan pada variabel Q. Namun apabila kondisi tersebut tidak terpenuhi, maka akan keluar dari seleksi kondisi tersebut.

Setelah mendapatkan titik Q, R dan S maka nilai dari variabel intervalQRS bisa didapatkan dengan menghitung selisih dari titik S dan titik Q atau bisa ditulis x[1][S]-x[1][Q]. Sementara nilai dari variabel gradienQRS didapatkan dengan cara membagi nilai variabel dacConv dengan nilai variabel DeltaX. DeltaY merupakan selisih antara nilai titik R dan titik Q secara vertikal, sedangkan DeltaX merupakan selisih antara nilai waktu R dan nilai waktu Q secara horizontal. Nilai DeltaY akan dikonversi berdasarkan Persamaan 5.1.

$$Vinput = \left(\frac{x}{2^n}\right) \times Vref \quad (5.1)$$

Keterangan:

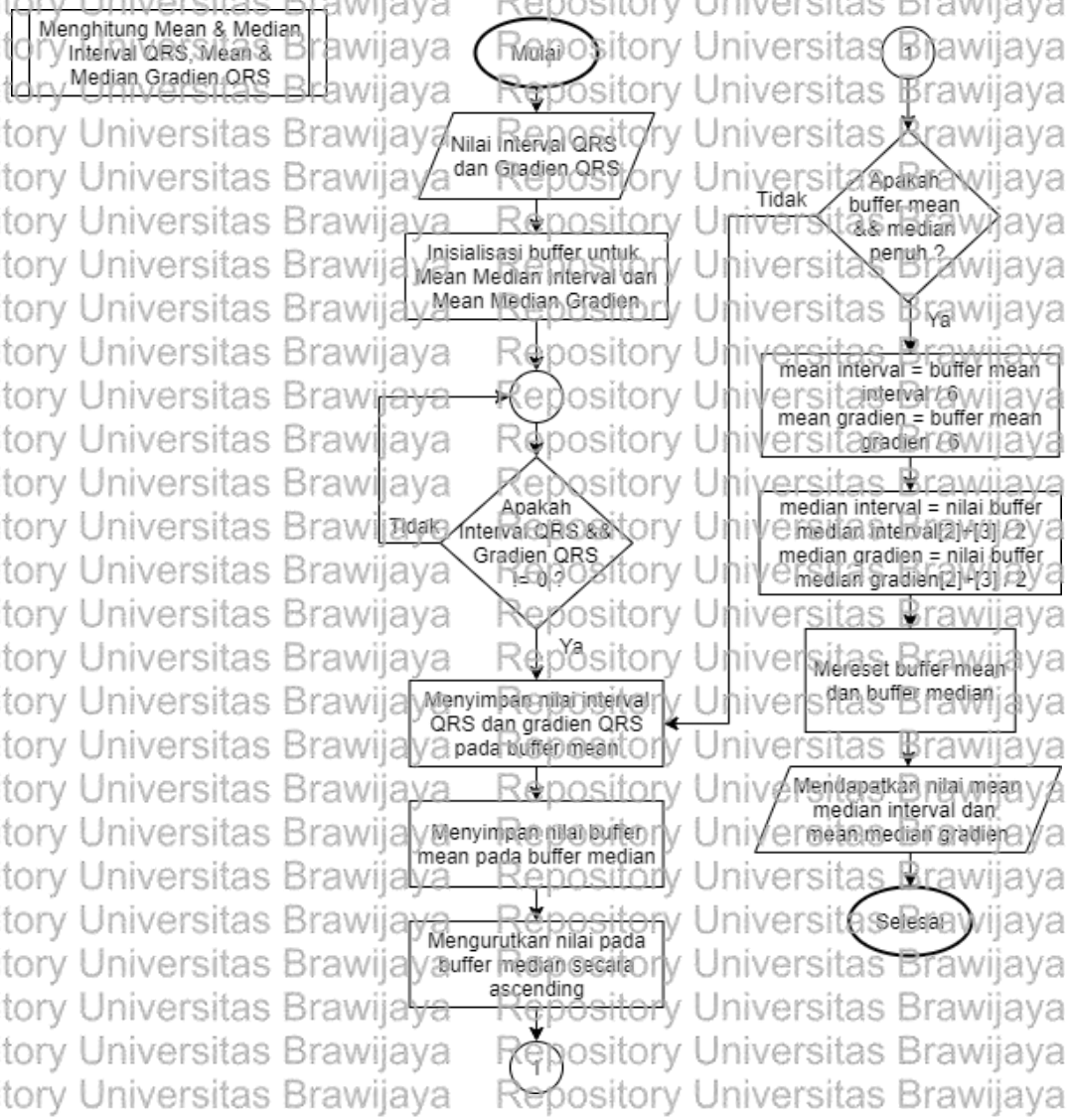
$x = nilai\ ADC\ yang\ ditampilkan$

$n = resolusi\ ADC\ Arduino$

$Vref = tegangan\ yang\ digunakan$

Nilai DeltaY perlu diubah karena nilai yang ditampilkan merupakan hasil *ADC Converter* yang terdapat pada Arduino. Hal tersebut perlu dilakukan untuk menyamakan rentang antara akuisisi sensor AD8232 dengan data yang diperoleh melalui *database online*. Sehingga nilai amplitude tetap dalam skala millivolt.

5.1.3.4 Perancangan Ekstraksi Fitur Mean dan Median



Gambar 5.8 Diagram Alir Ekstraksi Fitur Mean dan Median

Setelah mendapatkan nilai dari interval dan gradien QRS, maka diperlukan buffer sebagai penyimpanan sementara. Hal tersebut berfungsi untuk mendapatkan nilai dari fitur berpean sebagai masukan dari metode Jaringan Saraf Tiruan. Buffer yang digunakan adalah berupa array dan disiapkan sebanyak fitur yang digunakan, yaitu empat buffer. Nantinya setiap buffer akan menyimpan 6 siklus detak jantung yang kemudian akan dihitung rata-rata(mean) dan nilai tengahnya(median). Nilai rata-rata (mean) dan nilai tengah (median) dihitung berdasarkan rumus pada persamaan 5.2 dan persamaan 5.3.

$$\bar{X} = \frac{x_1+x_2+x_3 \ldots x_n}{n} \quad (5.2)$$

$$M_e = \frac{1}{2}\left(x_{\left(\frac{n}{2}\right)} + x_{\left(\frac{n}{2}+1\right)}\right) \quad (5.3)$$

Keterangan :

$M_e = Median$
$\bar{x} = Mean$
$x = urutan\ data$
$n = banyak\ data$

Untuk memasukkan nilai dari interval dan gradien QRS ke dalam buffer, sistem akan memeriksa apakah nilai dari kedua variabel tersebut tidak bernilai nol. Apabila kondisi tersebut tidak terpenuhi, maka sistem akan memeriksa kembali kondisi tersebut. Namun apabila kondisi terpenuhi, maka sistem akan memasuki kondisi perulangan. Pada kondisi perulangan tersebut nilai dari variabel interval QRS akan dimasukkan ke dalam buffer mean dan median interval QRS, sementara nilai dari variabel gradien QRS akan dimasukkan ke dalam buffer mean dan median gradien QRS. Kemudian pada buffer median interval QRS dan buffer median gradien QRS, setiap nilai yang masuk akan diurutkan dengan format ascending.

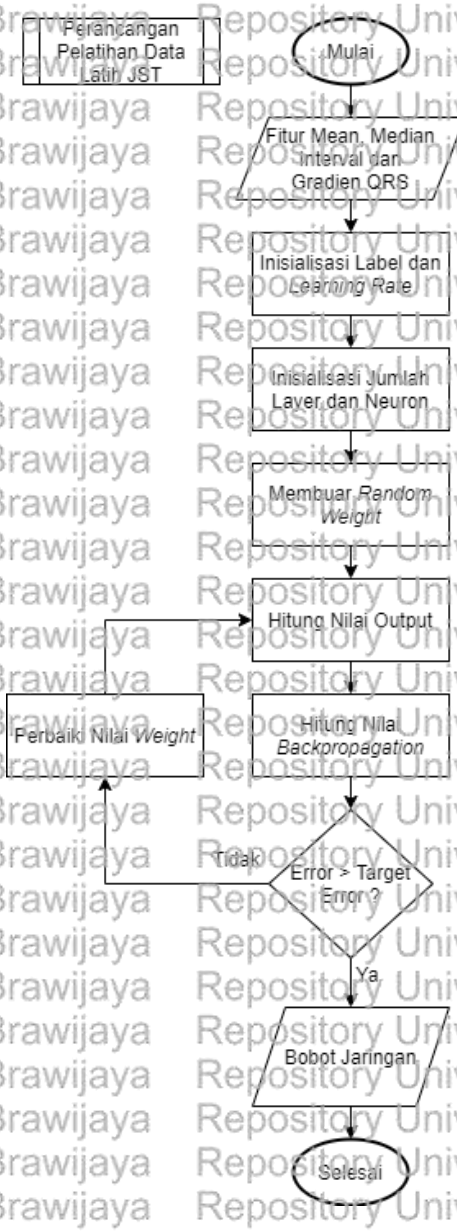
Setelah itu, sistem akan memeriksa apakah buffer mean median interval QRS dan buffer mean median gradien QRS sudah penuh. Apabila buffer tersebut sudah penuh, maka mean interval QRS dapat ditentukan dengan menjumlahkan semua nilai yang telah disimpan pada buffer mean interval QRS dan membaginya dengan nilai 6. Hal tersebut juga berlaku untuk mean gradien QRS, dimana nilai dari mean gradien QRS ditentukan dengan menjumlahkan semua nilai yang telah disimpan pada buffer mean gradien QRS dan membaginya dengan nilai 6. Nilai 6 digunakan karena banyaknya nilai interval maupun gradien yang disimpan adalah sebanyak 6 nilai.

Selain itu, nilai dari median interval QRS juga dapat ditentukan dengan mencari nilai tengah dari buffer median interval QRS. Karena nilai yang disimpan adalah sebanyak 6 nilai interval QRS, maka mediannya adalah nilai dari indeks ke-2 ditambah dengan nilai dari indeks ke-3 dari buffer median interval QRS kemudian dibagi 2. Hal tersebut juga berlaku untuk median gradien QRS, dimana nilai mediannya adalah nilai dari indeks ke 2 ditambah dengan nilai dari indeks ke-3 dari buffer median gradien QRS kemudian dibagi 2. Setelah mendapatkan mendeklarasikan mean median interval QRS dan mean median gradien QRS maka buffer akan direset kembali.

5.1.3.5 Perancangan Pelatihan Data Latih Metode Jaringan Saraf Tiruan

Gambar 5.9 merepresentasikan diagram alir untuk perancangan pelatihan data latih pada metode Jaringan Saraf Tiruan (JST). Proses pelatihan data latih akan dilakukan menggunakan Bahasa pemrograman python 3.9 dengan bantuan Pycharm. Algoritma *backpropagation* merupakan algoritma yang akan digunakan pada proses pelatihan ini. Tujuan dilakukannya pelatihan data ini adalah mencari nilai *weight* dan *bias* tetap pada metode JST. Untuk mendapatkan nilai *weight* dan

bias tetap, akan dilakukan iterasi sampai nilai error yang didapatkan sudah lebih kecil dari batas nilai error yang diinginkan.



Gambar 5.9 Diagram Alir Pelatihan Data Latih JST

Nilai fitur yang ditentukan berjumlah 40 data yang terbagi atas 20 data kondisi FA dan 20 data kondisi normal. Selanjutnya nilai fitur tersebut akan dijadikan *input* pada proses pelatihan ini. Sesuai dengan jumlah nilai fitur yang digunakan yaitu 4 fitur maka neuron yang terdapat pada *input* layer jaringan adalah sebanyak 4 neuron. Sementara itu banyaknya kelas yang akan ditentukan adalah 2 kelas, sehingga neuron pada *output* layer jaringan adalah sebanyak 2 neuron. Selanjutnya untuk menentukan jumlah neuron pada *hidden layer* digunakan Persamaan 2.4. Nilai *learning rate* yang digunakan adalah $10e^{-2}$. Nilai tersebut diambil karena hasil dari penelitian yang dilakukan oleh (Siregar, 2019) dan (Siregar, 2017) dengan menggunakan nilai *learning rate* $10e^{-2}$ mendapatkan hasil akurasi yang cukup tinggi. Sedangkan untuk menentukan nilai *epoch* dilakukan

percobaan dengan target nilai *error* dari *loss function cross entropy* sebesar 0.01. Tabel 5.4 menunjukkan pengaruh nilai *epoch* terhadap target nilai *error* untuk fase pelatihan.

Tabel 5.4 Pengaruh Nilai *epoch* Pada Pelatihan JST

No	*Epoch*	Waktu (*s*)	*Error loss*
1	20000	1,820	1.8262
2	50000	4,469	0.0436
3	80000	7,016	0.0136

Berdasarkan Tabel 5.4, nilai *epoch* yang akan digunakan adalah 80000. Nilai tersebut dapat mencapai nilai error target yang diinginkan, yaitu 0.0136 dengan waktu pelatihan 7,016 sekon. Kemudian menginisialisasi *random weight* sebagai bobot awal proses pelatihan. Proses pelatihan dimulai dengan fase *feedforward* yang dilakukan sesuai Persamaan 2.8, 2.9, 2.10 dan 2.11 dengan melibatkan fungsi aktivasi sigmoid pada *hidden layer* dan fungsi aktivasi softmax pada *output layer* sesuai Persamaan 2.5 dan Persamaan 2.6 untuk mendapatkan nilai *output* jaringan. Apabila nilai *output* jaringan yang telah didapatkan masih lebih besar dari nilai *error* yang diinginkan maka proses pelatihan memasuki fase *backpropagation.* Pada fase *backpropagation* nilai *output* tersebut akan melalui proses perhitungan sesuai dengan Persamaan 2.13, 2.14, 2.15, 2.16, dan 2.17. Setelah itu, proses pelatihan akan memasuki fase terakhir yaitu *updating weight* yang dilakukan berdasarkan Persamaan 2.18, 2.19, 2.20 dan 2.21. Proses pelatihan nantinya akan berhenti setelah 80000 *epoch* atau 80000 iterasi.

5.1.3.6 Perancangan Klasifikasi Menggunakan JST

Gambar 5.10 merepresentasikan diagram alir dari klasifikasi Jaringan Saraf Tiruan yang akan diimplementasikan pada Arduino UNO. Nilai fitur yang telah didapatkan akan dinormalisasi terlebih dahulu untuk menyamakan skala nilai tersebut dengan data latih. Setelah hal tersebut dilakukan maka tahap selanjutnya adalah memulai perhitungan klasifikasi berdasarkan subbab 2.2.6 dengan menggunakan nilai *weight* dan *epoch* yang didapatkan dari proses pelatihan sebelumnya. Model JST pada penelitian ini adalah dengan menggunakan 4 neuron pada lapisan masukan sesuai dengan fitur yang digunakan. Sedangkan untuk neuron pada *output layer* adalah sebanyak 2 neuron sesuai dengan kelas yang ingin diklasifikasikan, yaitu kelas "normal" dan kelas "FA".

Gambar 5.10 Diagram Alir Klasifikasi Jaringan Saraf Tiruan

Melalui perhitungan pada Persamaan 2.3 dapat ditentukan jumlah neuron yang terdapat pada *hidden layer*, yaitu:

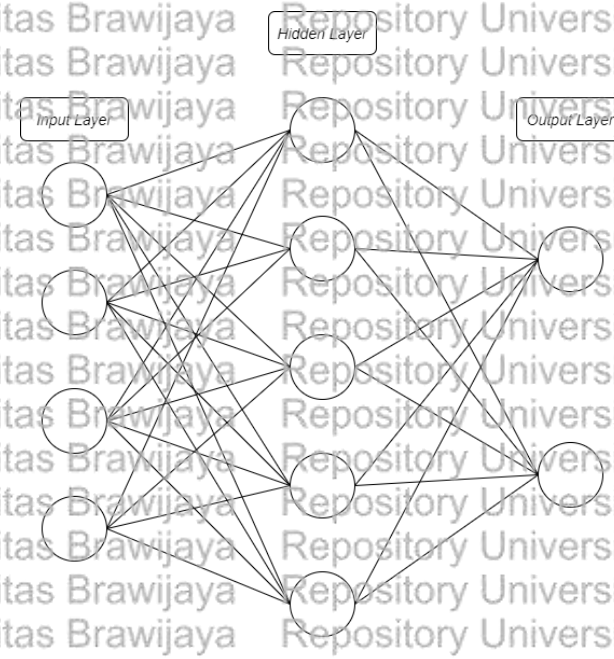
$$\sum hN = round(\frac{24}{3} + 2) \quad (2.3)$$

$$\sum hN = round(2.66 + 2)$$

$$\sum hN = round(4.66)$$

$$\sum hN = 5$$

Nilai 4 merepresentasikan jumlah fitur yang digunakan, yaitu mean median dari interval QRS dan mean median dari gradien QRS. Sementara nilai 2 merepresentasikan kelas yang ingin diklasifikasikan, yaitu kelas "Normal" dan kelas "FA". Sehingga didapatkan jumlah neuron pada hidden layer adalah 5. Setelah mendapatkan jumlah neuron pada masing-masing layer, maka model arsitektur JST dari sistem telah ditemukan. Gambar 5.11 merepresentasikan model JST dari sistem yang dibangun.



Gambar 5.11 Model Arsitektur Jaringan Saraf Tiruan pada Sistem

Pada proses pelatihan yang telah dilakukan sebelumnya menghasilkan nilai *weight* dan nilai *bias* yang terdapat pada *hidden layer* maupun *output layer*. Nilai *weight* dan *bias* untuk masing-masing *layer* sendiri ditunjukkan pada Tabel 5.5 dan Tabel 5.6.

Tabel 5.6 Nilai *weight hidden layer* Hasil Pelatihan

weight / *node*	wh_1	wh_2	wh_3	wh_4	bh
$node_1$	-10,2711	-7,6717	-10,3431	-20,9245	6,9244
$node_2$	56,1754	-0,4006	-0,3256	28,0562	-30,2436
$node_3$	-34,8480	8,4843	-2,7394	39,7695	-8,8275
$node_4$	8,8106	16,3244	-13,4891	19,0348	15,7928
$node_5$	40,6057	12,4849	-2,2909	-10,0715	4,0378

Tabel 5.7 Nilai *weight output layer* Hasil Pelatihan

weight / *node*	wo_1	wo_2	wo_3	wo_4	wo_5	bo
$node_1$	-21,2230	24,3025	-24,3321	-12,1680	11,8357	13,0178
$node_2$	21,8410	-22,9736	25,3877	13,4031	-10,2378	-13,0597

Untuk melakukan perhitungan manual, digunakan salah satu data uji dengan nilai fitur rata-rata dan median dari interval QRS serta rata-rata dan median dari gradien QRS berturut-turut adalah 60,67, 62, 10,64 dan 10,51. Keempat fitur

tersebut pertama-tama akan masuk ke tahap normalisasi data sesuai dengan Persamaan 2.4. Perhitungan normalisasi ditunjukkan dalam Tabel 5.8

Tabel 5.8. Normalisasi Nilai Fitur

Interval QRS		Gradien QRS	
Mean	Median	Mean	Median
$D_{normal} = \frac{D - D_{min}}{D_{max} - D_{min}}$	$D_{normal} = \frac{D - D_{min}}{D_{max} - D_{min}}$	$D_{normal} = \frac{D - D_{min}}{D_{max} - D_{min}}$	$D_{normal} = \frac{D - D_{min}}{D_{max} - D_{min}}$
$D_{normal} = \frac{60{,}67 - 46{,}33}{105{,}33 - 46{,}33}$	$D_{normal} = \frac{60 - 46{,}50}{102 - 46{,}50}$	$D_{normal} = \frac{10{,}64 - 7{,}44}{48{,}86 - 7{,}44}$	$D_{normal} = \frac{10{,}51 - 7{,}41}{48{,}89 - 7{,}41}$
$D_{normal} = \frac{13}{59}$	$D_{normal} = \frac{15.5}{55{,}5}$	$D_{normal} = \frac{3.2}{41{,}42}$	$D_{normal} = \frac{3.1}{41.48}$
$D_{normal} = 0{,}2431$	$D_{normal} = 0{,}2793$	$D_{normal} = 0{,}0773$	$D_{normal} = 0{,}0747$

Setelah melakukan normalisasi terhadap keempat fitur sesuai dengan Tabel 5.8, tahap selanjutnya adalah melakukan perhitungan *weight (zh)* untuk seluruh *node* pada *hidden layer* yang berjumlah 5 *node* dengan menggunakan Persamaan 2.8. Proses perhitungan dan *weight (zh)* ditunjukkan dalam Tabel 5.9.

Tabel 5.9 Perhitungan *weight zh* pada *Hidden Layer*

neuron	*weight zh*
1	$zh_1 = (-10{,}2711 \cdot 0{,}2431) + ((-7{,}6717) \cdot 0{,}2793) + ((-10{,}3431) \cdot 0{,}0773) + ((-20{,}9245) \cdot 0{,}0747) + (6{,}9244)$ $zh_1 = -0{,}0778$
2	$zh_2 = (56{,}1754 \cdot 0{,}2431) + ((-0{,}4006) \cdot 0{,}2793) + ((-0{,}3256) \cdot 0{,}0773) + (28{,}0562 \cdot 0{,}0747) + (-30{,}2436)$ $zh_2 = -14{,}6286$
3	$zh_3 = ((-34{,}8480) \cdot 0{,}2431) + (8{,}4843 \cdot 0{,}2793) + ((-2{,}7394) \cdot 0{,}0773) + (39{,}7695 \cdot 0{,}0747) + (-8{,}8275)$ $zh_3 = -12{,}1704$
4	$zh_4 = (8{,}8106 \cdot 0{,}2431) + (16{,}3244 \cdot 0{,}2793) + ((-13{,}4891) \cdot 0{,}0773) + (19{,}0348 \cdot 0{,}0747) + (15{,}7928)$ $zh_4 = 22{,}7940$
5	$zh_5 = (40{,}6057 \cdot 0{,}2431) + (12{,}4849 \cdot 0{,}2793) + ((-2{,}2909) \cdot 0{,}0773) + ((-10{,}0715) \cdot 0{,}0747) + (4{,}0378)$ $zh_5 = 16{,}4189$

Selanjutnya nilai *weight zh* seluruh *node* pada *hidaen layer* yang telah didapatkan akan dilakukan perhitungan kembali dengan fungsi aktivasi sigmoid. Perhitungan tersebut dilakukan sesuai dengan Persamaan 2.9 dan menghasilkan nilai *weight ah* dalam Tabel 5.10.

Tabel 5.10 Perhitungan *weight ah* pada *Hidden Layer*

neuron	*weight ah*
1	$zh_1 = -0{,}0778$ $ah_1 = sigmoid(-0{,}0778)$ $ah_1 = 0{,}4806$
2	$zh_2 = -14{,}6286$ $ah_2 = sigmoid(-14{,}6286)$ $ah_2 = 0{,}0000$
3	$zh_3 = -12{,}1704$ $ah_3 = sigmoid(-12{,}1704)$ $ah_3 = 0{,}0000$
4	$zh_4 = 22{,}7940$ $ah_4 = sigmoid(22{,}7940)$ $ah_4 = 1{,}0000$
5	$zh_5 = 16{,}4189$ $ah_5 = sigmoid(16{,}4189)$ $ah_5 = 1{,}0000$

Nilai *weight ah* pada *hidden layer* yang telah didapatkan menandakan perhitungan pada *hidden layer* telah selesai. Selanjutnya nilai tersebut akan digunakan untuk mendapatkan nilai *weight zo* pada *output layer* melalui perhitungan sesuai dengan Persamaan 2.10. Perhitungan nilai *weight zo* pada *output layer* ditunjukkan dalam Tabel 5.11.

Tabel 5.11 Perhitungan *weight zo* pada *Output Layer*

neuron	*weight zo*
1	$zo_1 = ((-21{,}2230) \cdot 0{,}4806) + (24{,}3025 \cdot 0{,}000) + ((-24{,}3321) \cdot 0{,}0000) + ((-12{,}1680) \cdot 1{,}0000) + (11{,}8357 \cdot 1{,}0000) + (13{,}0178)$ $zo_1 = 2{,}4857$
2	$zo_2 = (21{,}8410 \cdot 0{,}4806) + ((-22{,}9736) \cdot 0{,}0000) + (25{,}3877 \cdot 0{,}0000) + (13{,}4031 \cdot 1{,}0000) + ((-10{,}2378) \cdot 1{,}0000) + (-13{,}0597)$ $zo_2 = 0{,}6024$

Selanjutnya nilai *weight zo* yang telah didapatkan akan dilakukan perhitungan kembali dengan fungsi aktivasi softmax. Perhitungan tersebut dilakukan berdasarkan Persamaan 2.11 dan menghasilkan nilai *weight ao* dalam Tabel 5.12.

Tabel 5.12 Perhitungan *weight ao* pada *Output Layer*

neuron	*weight ah*
1	$zo_1 = 2{,}4857$ $ao_1 = softmax(2{,}4857)$ $ao_1 = 0{,}8680$
2	$zo_2 = 0{,}6024$ $ao_2 = softmax(0{,}6024)$ $ao_2 = 0{,}1320$

Nilai *weight ao* pada *output layer* yang telah didapatkan adalah 0,8680 untuk ao_1 dan 0,1320 untuk ao_2. Nilai *weight* tersebut merepresentasikan kelas yang ditentukan, yaitu kelas "FA" dan kelas "normal". Kemudian untuk menentukan data uji tersebut termasuk ke dalam kelas yang mana dapat ditentukan dengan mengambil nilai ao terbesar pada *output layer.* Sesuai dengan Tabel 5.11 nilai ao terbesar adalah pada ao_1 dengan nilai 0,8680. Oleh karena itu, data uji dengan nilai fitur mean interval QRS = 59,33, median interval QRS = 58, mean gradien QRS = 11,80 dan median gradien QRS = 11,88 setelah dilakukan klasifikasi menggunakan metode JST termasuk kedalam kelas FA.

5.2 Implementasi Sistem

Melalui perancangan yang telah dilakukan, maka sub bab ini berisi tentang proses implementasi dari perancangan tersebut. Sub bab implementasi sistem sendiri terdiri atas implementasi *packaging* sistem, implementasi perangkat keras dan implementasi perangkat lunak.

5.2.1 Implementasi *Packaging* Sistem

Implementasi pertama merupakan implementasi dari *packaging* sistem berdasarkan perancangan *packaging* untuk sistem pendeteksi Fibrilasi Atrium (FA) yang dilakukan pada sub bab perancangan *packaging* sistem. Gambar 5.11 merepresentasikan implementasi dari *packaging* sistem.

Pada implementasi *packaging* sistem pendeteksi Fibrilasi Atrium (FA), digunakan bahan material akrilik sebagai pengemasan dari perangkat kerasnya. Akrilik yang digunakan berwarna hitam dan dibentuk menyerupai balok dengan ukuran 14 cm x 10 cm x 8,5 cm. Di dalam *packaging* terdapat perangkat keras yang membangun sistem ini, diantaranya Arduino Uno, Sensor AD8232, LCD 16x2, *Breadboard* dan Kabel *Jumper*.

Gambar 5.12 Implementasi Packaging Sistem

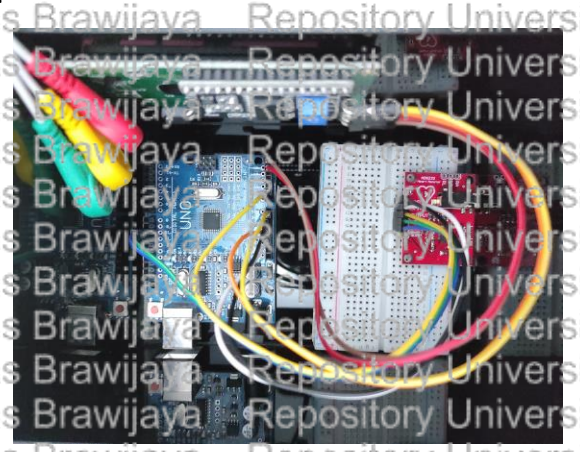
5.2.2 Implementasi Perangkat Keras

Implementasi kedua adalah implementasi dari perangkat keras sistem menurut perancangan perangkat keras yang telah dilakukan pada sub bab perancangan perangkat keras dari sistem pendeteksi Fibrilasi Atrium (FA).

5.2.2.1 Implementasi Perangkat Keras Mikrokontroler

Pada sub bab ini akan terdapat penjelasan implementasi dari perancangan perangkat keras mikrokontroler yang terdiri dari Arduino Uno, Sensor AD8232, Breadboard dan LCD 16x2 pada sub bab perancangan perangkat keras mikrokontroler. Hasil implementasi sendiri dapat dilihat pada Gambar 5.13.

Setiap komponen dihubungkan sesuai dengan pin yang telah ditentukan pada sub bab perancangan perangkat keras mikrokontroler. Untuk menghubungkan komponen-komponen tersebut digunakan kabel *jumper male female* dan *male male*. Pada penghubungan Arduino Uno dengan Sensor AD8232, digunakan breadboard untuk membuat Sensor AD8232 lebih kokoh dan memudahkan pemasangan. Sehingga kabel *jumper* yang digunakan adalah jenis *male male*. Sementara pada penghubungan Arduino dengan LCD 16x2, digunakan Modul I2C untuk menghemat pemakaian pin pada Arduino. Sehingga pin yang digunakan hanya 4 buah termasuk *power supply* dan kabel *jumper* yang digunakan adalah jenis *male female*.



Gambar 5.13 Implementasi Perangkat Keras Mikrokontroler

5.2.2.2 Implementasi Penempatan Elektrode

Pada bagian ini akan terdapat penjelasan implementasi dari peletakkan elektrode pada sistem pendeteksi Fibrilasi Atrium yang diletakkan pada tubuh. Penempatan elektrode dilakukan berdasarkan perancangan sub bab sebelumnya. Gambar 5.14 merepresentasikan peletakkan elektrode pada subjek.

Gambar 5.14 Implementasi Penempatan Elektrode

Sesuai dengan Gambar 5.14, elektrode telah diletakkan sesuai dengan tahap perancangan. Elektrode berwarna merah ditempelkan pada dada sebelah kanan, elektrode berwarna kuning ditempelkan pada dada sebelah kiri dan elektrode berwarna hijau ditempelkan pada perut kanan bagian bawah.

5.2.3 Implementasi Perangkat Lunak

Pada bagian ini akan dijelaskan tentang proses implementasi yang dilakukan terhadap perancangan yang telah dilakukan pada sub bab perancangan perangkat lunak untuk sistem pendeteksi Fibrilasi Atrium (FA). Implementasi perangkat lunak sendiri terbagi menjadi enam sub bahasan, antara lain implementasi program utama, implementasi pengambilan data melalui sensor AD8232, implementasi ekstraksi fitur interval dan gradien QRS, implementasi ekstraksi fitur rata-rata dan median, implementasi pelatihan data latih JST dan implementasi klasifikasi menggunakan JST.

5.2.3.1 Implementasi Program Utama

Pada bagian ini akan dijelaskan mengenai kode program utama yang membangun sistem pendeteksi Fibrilasi Atrium (FA). Penjelasan kode program akan dibagi menjadi beberapa bagian sesuai dengan komponen yang membangun sistem maupun fungsi perhitungan yang diperlukan untuk mengolah data.

Tabel 5.13 Kode Program Inisialisasi Komponen dan Variabel

No	*Source Code*
1	`#include <LiquidCrystal_I2C.h>`
2	`#include <Wire.h>`
3	
4	`#define PIN_LO_minus 10`
5	`#define PIN_LO_plus 11`
6	`#define PIN_output A0`
7	
8	`int x[2][15];`
9	`int inputSignal = 0;`
10	`unsigned long waktu, time1;`
11	`int Q = 0;`
12	`int R = 0;`
13	`int S = 0;`
14	`int intervalQRS = 0;`
15	`float DeltaX, DeltaY, dacConv, gradienQRS = 0;`
16	`const int periode = 7;`
17	`float buffer_mean_iQRS[periode];`
18	`float buffer_med_iQRS[periode];`
19	`float buffer_mean_gQRS[periode];`
20	`float buffer_med_gQRS[periode];`
21	
22	`LiquidCrystal_I2C lcd(0x27, 16, 2);`

Tabel 5.13 merupakan kode program untuk menginisialisasi pin, variabel dan komponen yang dibutuhkan. Baris 1-2 merupakan inisialisasi untuk menggunakan *library* LCD 16x2 *with* I2C. Baris 4-6 merupakan inisialisasi pin yang digunakan oleh sensor AD8232 dalam mendeteksi dan mengirimkan sinyal EKG pada subjek. Selanjutnya baris 8-14 merupakan inisialisasi variabel yang digunakan dalam pendeteksian fitur yang digunakan, merupakan inisialisasi untuk variabel yang dibutuhkan untuk mengolah data hasil sinyal EKG dari subjek. Dan yang terakhir adalah inisialisasi alamat, banyak kolom dan baris pada komponen LCD yang digunakan.

Tabel 5.14 Kode Program Inisialisasi Model Arsitektur JST

No	*Source Code*
1	`struct JST{`
2	`  float wH[4][5] = {`
3	`    {-10.2710903845,  56.1753851258,  -34.8480049640,`
4	`8.8105703259, 40.6057061173},`
5	`    {-7.6716534339,  -0.4006453514,  8.4842810977,`
6	`16.3243986601, 12.4849211983},`
7	`    {-10.3430855690,  -0.3255686300,  -2.7394381673,  -`
8	`13.4890648340, -2.2909478715},`
9	`    {-20.9245183818,  28.0562463906,  39.7694681736,  -`
10	`19.0347480833, -10.0715154545}`
11	`  };`
12	
13	`  float bH[5] = {`
14	`    6.9244434205,  -30.2436169223,  -8.8274968002,`
15	`15.7928060057, 4.0377848536`
16	`  };`
17	
18	`  float wO[5][2] = {`

```
19      {-21.2230022363, 21.8410093066},
20      {24.3024860354, -22.9735736447},
21      {-24.3320534396, 25.3877183513},
22      {-12.1679888766, 13.4030733071},
23      {11.8356704115, -10.2378431745}
24    };
25
26    float bO[2] = {
27      13.0178455700, -13.0597468791
28    };
29
30    float sigF(float x){
31      return (float)1/(1+expf(-x));
32    };
33
34    float softF(float *array, int i){
35      static float jumlahExp = 0;
36      if(jumlahExp == 0){
37        for(int j = 0; j<6; j++){
38          jumlahExp += expf(array[j]);
39        }
40      }
41      return expf(array[i]) / jumlahExp;
42    };
43
44    float lHidden[5], lOutput[2];
45    const String kelas[2] = {"FA", "Normal"};
46    float timeCalc, fWeight;
47  };
48  struct JST jst;
```

Tabel 5.14 merepresentasikan model arsitektur JST yang digunakan pada penelitian ini. Baris 2-16 merupakan implementasi dari nilai *weight* dan *bias* dari *hidden layer*. Sedangkan baris 18-28 merupakan implementasi dari nilai *weight* dan *bias* dari *output layer*. Kemudian implementasi dari fungsi aktivasi yang dipakai, yaitu fungsi aktivasi sigmoid dan fungsi aktivasi *softmax* terdapat pada baris 30-42. Sementara itu, baris 44-46 merupakan inisialisasi variabel dari jumlah *hidden layer* yang digunakan yaitu 5, jumlah *output layer* yang digunakan yaitu 2, nama kelas yang digunakan yaitu "FA" dan "Normal" serta variabel untuk menyimpan waktu komputasi dan bobot dari hasil deteksi nantinya.

Tabel 5.15 Kode Program Normalisasi dan Pengurutan Data

No	*Source Code*
1	`struct dataUji{`
2	`  float mean_iQRS, median_iQRS, mean_qQRS, median_qQRS;`
3	`  char kelas = '-';`
4	`};`
5	
6	`struct dataUji data_uji;`
7	
8	`void normalisasi_data(dataUji &data_uji){`
9	`  data_uji.mean_iQRS = (data_uji.mean_iQRS - 46.33)/(105.33`
10	`- 46.33);`
11	`  data_uji.median_iQRS = (data_uji.median_iQRS -`
12	`46.50)/(102.00 - 46.50);`
13	

```
14    data_uji.mean_gQRS = (data_uji.mean_gQRS - 7.44)/(48.86 -
15  7.44);
16    data_uji.median_gQRS     =     (data_uji.median_gQRS     -
17  7.41)/(48.89 - 7.41);
18  }
19
20  void urutkanMedian(float *dataArry, byte jumlahData){
21    byte i = 0;
22    for (byte i=0; i<jumlahData; i++){
23      byte k;
24      for (byte j=1; j<jumlahData; j++){
25        if(dataArray[j-1] > dataArray[j]){
26          float temp = dataArray[j-1];
27          dataArray[j-1] = dataArray[j];
28          dataArray[j] = temp;
29        }
30      }
31    }
  }
```

Tabel 5.15 merupakan implementasi dari pembuatan struktur data untuk data yang akan diujikan, fungsi normalisasi dan fungsi pengurutan yang dibutuhkan untuk mengolah data sinyal EKG pada subjek. Pada baris 1-4 merupakan struktur data uji yang disiapkan. Didalamnya terdapat variabel untuk menyimpan fitur yang digunakan beserta variabel kelas untuk menyimpan hasil pendeteksian kelasnya. Kemudian baris 8-17 merupakan fungsi normalisasi yang bertujuan untuk menyamakan rentang nilai dari data uji yang didapatkan sebelum dilakukan klasifikasi dengan model JST yang sudah dibuat sebelumnya. Sementara pada baris 19-31 merupakan fungsi untuk mengurutkan nilai dari fitur yang telah didapatkan. Fungsi tersebut digunakan untuk mendapatkan nilai median dari interval QRS dan nilai median dari gradien QRS.

Tabel 5.16 Kode Program *Void Setup* pada Arduino

No	*Source Code*
1	`Void setup() {`
2	`  Serial.begin(9600);`
3	`  pinMode(PIN_LO_minus, INPUT);`
4	`  pinMode(PIN_LO_plus, INPUT);`
5	`  lcd.init();`
6	`  lcd.backlight();`
7	`  lcd.setCursor(0,0);`
8	`  lcd.print("   DETEKSI");`
9	`  lcd.setCursor(0,1);`
10	`  lcd.print("        FA");`
11	`  delay(1000);`
12	`  lcd.clear();`
13	`}`

Tabel 5.16 merepresentasikan fungsi *setup* pada sistem ini. Fungsi ini hanya akan dijalankan sekali saat sistem menyala. Fungsi ini berisi inisialisasi boudrate yang digunakan, yaitu 9600 sesuai dengan baris 2. Inisialisasi PIN_LO_minus dan PIN_LO_plus pada sensor AD8232 sebagai pin yang menerima input sesuai dengan baris 3-4 serta inisialisasi komponen LCD 16x2 pada saat sistem menyala dengan

menampilkan "DETEKSI FA" selama 1 detik. Sedangkan untuk proses lainnya yang terdapat pada fungsi *loop* akan dibahas sesuai pada blok program pada perancangan program utama.

5.2.3.2 Implementasi Pengambilan Data Melalui Sensor AD8232

Pada bagian ini terdapat penjelasan kode program untuk pengambilan data melalui sensor AD8232 yang ditunjukkan dalam Tabel 5.17.

Tabel 5.17 Kode Program Pengambilan Data melalui Sensor AD8232

No	*Source Code*
1	`void loop() {`
2	`  inputSignal = analogRead(PIN_output);`
3	`  waktu = millis();`
4	`  if((analogRead(PIN_LO_minus) ==`
5	`1)||(analogRead(PIN_LO_plus) == 1)){`
6	`    Serial.print('!');`
7	`  }else{`
8	`    Serial.println(inputSignal);`
9	`  }`

Sesuai dengan Tabel 5.17, Nilai data yang dideteksi oleh sensor akan dikirim ke Arduino dan disimpan pada variabel inputSignal sesuai dengan baris 2. Sementara itu, fungsi millis yang berguna untuk menjalankan waktu secara independen akan disimpan pada variabel waktu sesuai dengan baris 3. Hal tersebut akan berguna untuk dapat menghitung interval dan gradien QRS nantinya. Kemudian bagian kode seleksi kondisi pada baris 4-8 merupakan proses untuk mendeteksi status pin LO+ dan LO- pada sensor AD8232. Apabila salah satu dari kedua pin tersebut bernilai 1 (HIGH) maka terindikasi bahwa elektrode belum dapat membaca nilai data jantung. Namun apabila kondisi tersebut tidak terpenuhi, maka nilai data yang terdeteksi akan ditampilkan pada serial monitor. Nilai data tersebut merepresentasikan aktivitas listrik pada jantung dan bentuk sinyal jantungnya dapat dilihat pada serial plotter.

5.2.3.3 Implementasi Ekstraksi Fitur Interval dan Gradien QRS

Pada sub bab ini akan dijelaskan mengenai kode program untuk mendapatkan nilai interval dan gradien QRS pada sistem pendeteksi Fibrilasi Atrium (FA). Tabel 5.18 merepresentasikan kode programnya.

Tabel 5.18 Kode Program Ekstraksi Fitur Interval dan Gradien QRS

No	Kode Program
1	`for(int i=14; i>=0; i--){`
2	`  if(i==0){`
3	`    x[0][i] = inputSignal;`
4	`    x[1][i] = waktu;`
5	`    continue;`
6	`  }`
7	`  x[0][i] = x[0][i-1];`
8	`  x[1][i] = x[1][i-1];`
9	`}`
10	

```
11    if ((x[0][1]>420&&x[0][1]>x[0][0]&&x[0][1]>x[0][2]) ||
12  R>0){
13      R++;
14      if(x[0][0]>x[0][1]&&x[0][1]<x[0][R]){
15        S++;
16      }
17    }
18
19  if (S>0){
20      for(int i=R; i<15; i++){
21        if(x[0][i]>=x[0][i-1]&&x[0][i-1]<x[0][i-2]){
22          Q = i-1;
23        }
24      }
25      intervalQRS = x[1][S] - x[1][Q];
26      DeltaY = x[0][R] - x[0][Q];
27      DeltaX = x[1][R] - x[1][Q];
28      dacConv = ((DeltaY*3300)/1024);
29      gradienQRS = dacConv/DeltaX;
```

Sesuai dengan Tabel 5.18, bagian kode pada baris 1 – 9 merupakan fungsi perulangan yang berguna untuk menyimpan nilai data dan nilai waktu yang dideteksi oleh sensor. Kedua nilai tersebut akan dimasukkan ke dalam array x[2][15] yang telah diinisialisasi sebelumnya. Nilai data yang disimpan pada variabel inputSignal akan dimasukkan ke dalam array x[0][i] sementara nilai waktu yang disimpan pada variabel pewaktuan akan dimasukkan ke dalam array x[1][i].

Kemudian bagian kode pada baris 11 – 24 berguna untuk mendapatkan nilai dari titik Q, R dan S. Hal tersebut dilakukan dengan memakai seleksi kondisi *if*. Karena tiitk R merupakan titik tertinggi dari satu siklus sinyal jantung maka pertama-tama akan mencari titik R. Apabila Titik R telah ditemukan, maka akan masuk pada seleksi kondisi *if* yang kedua untuk mencari Titik S. Titik S sendiri merupakan titik yang paling rendah setelah titik R. Setelah itu dari Titik S akan dicari Titik Q dengan menggunakan referensi titik R. Apabila indeks telah melewati titik R, maka nilai terendah sebelum titik R merupakan titik Q. Nantinya Titik Q, R dan S akan digunakan untuk mendapatkan nilai dari interval dan gradien QRS. Setelah mendapatkan nilai dari titik Q, R dan S, maka bagian kode pada baris 25 – 29 berguna untuk mendapatkan nilai interval dan gradien QRS. Pada Tabel 5.19 menunjukkan jumlah data yang disimpan untuk penentuan interval QRS pada subjek pertama.

Tabel 5.19 Enam Siklus Data Interval QRS Subjek Pertama

No	Siklus 1	Siklus 2	Siklus 3	Siklus 4	Siklus 5	Siklus 6
1	280	335	255	168	266	252
2	248	335	212 (Q)	159 (Q)	209	207
3	193	326	239	369	143 (Q)	159 (Q)
4	113 (Q)	290	411	614	227	200
5	155	233	565	642 (R)	494	395
6	425	169 (Q)	567 (R)	478	656 (R)	553 (R)
7	648	272	437	321	588	538

8	620 (R)	546	349	252	406	393
9	434	659 (R)	312	239	288	306
10	283	606	291	235	248	268
11	226	428	279	233	235	246
12	215	301	274	233	232	230
13	212	258	272	231	230	223
14	209 (S)	250 (S)	272 (S)	231 (S)	230 (S)	223 (S)
15	210	251	274	233	232	228

Sesuai dengan Tabel 5.19, 15 baris yang digunakan untuk menyimpan jumlah data dari hasil deteksi sensor sudah mencukupi untuk dapat menghitung interval dan gradien QRS. Sementara itu Tabel 5.20 menunjukkan jumlah data interval QRS untuk subjek 2.

Tabel 5.20 Enam Siklus Data interval QRS Subjek Kedua

No	Siklus 1	Siklus 2	Siklus 3	Siklus 4	Siklus 5	Siklus 6
1	323	328	329	329	331	334
2	329	325	330	331	326	328 (Q)
3	331	323	331	332	322	329
4	318	318 (Q)	320	323	320	331
5	315 (Q)	319	311 (Q)	310 (Q)	315 (Q)	320
6	338	338	327	315	322	318
7	390	386	378	344	360	339
8	460	459	452	397	430	390
9	525	526	520	465	507	459
10	540 (R)	527 (R)	530 (R)	527 (R)	536 (R)	522
11	479	462	470	516	484	529 (R)
12	400	398	402	438	411	466
13	350	353	357	376	362	396
14	349 (S)	347 (S)	346 (S)	344 (S)	345 (S)	352 (S)
15	355	351	352	347	348	353

Dapat dilihat pada Tabel 5.20 jumlah data untuk interval QRS berada pada array ke 14 sama seperti Tabel 5.19. Oleh karena itu, baris array yang digunakan adalah sebanyak 15 baris. Setiap data yang berhasil di deteksi oleh sensor AD8232 akan dimasukkan ke dalam array dan setiap data juga akan ditentukan apakah data tersebut merupakan titik Q, R ataupun S. Tabel 5.6 mereperesentasikan data hasil deteksi sensor beserta waktu pada saat nilai dari data tersebut terdeteksi yang dimasukkan pada array [2, 15].

Tabel 5.21 Data Beserta Waktu Hasil Deteksi Sensor

No	Data	Waktu (ms)
1	255	4312
2	212	4317
3	239	4322

4	411	4327
5	565	4332
6	567	4337
7	437	4342
8	349	4347
9	312	4353
10	291	4358
11	279	4363
12	274	4368
13	272	4374
14	272	4379
15	274	4384

Setelah data hasil deteksi sensor beserta waktunya dimasukkan ke dalam array, maka dapat ditentukan nilai dari titik Q, R dan S yang ditunjukkan dalam Tabel 5.21

Tabel 5.22 Nilai dan Waktu Titik Q, R dan S untuk Data Tabel 5.21

Titik	Data	Waktu
Q	212	4317
R	567	4337
S	272	4379

Setelah mendapatkan nilai dari titik Q, R dan S, selanjutnya dapat dilakukan penentuan nilai interval QRS dan gradien QRS. Kedua nilai tersebut akan disimpan ke dalam buffer untuk langkah selanjutnya yang akan dijelaskan pada sub bab berikutnya. Untuk mendapatkan nilai interval QRS digunakan rumus sesuai dengan Persamaan 2.1.

$$Interval\ QRS = xS - xQ \tag{2.1}$$
$$= 4379 - 4317$$
$$= 62$$

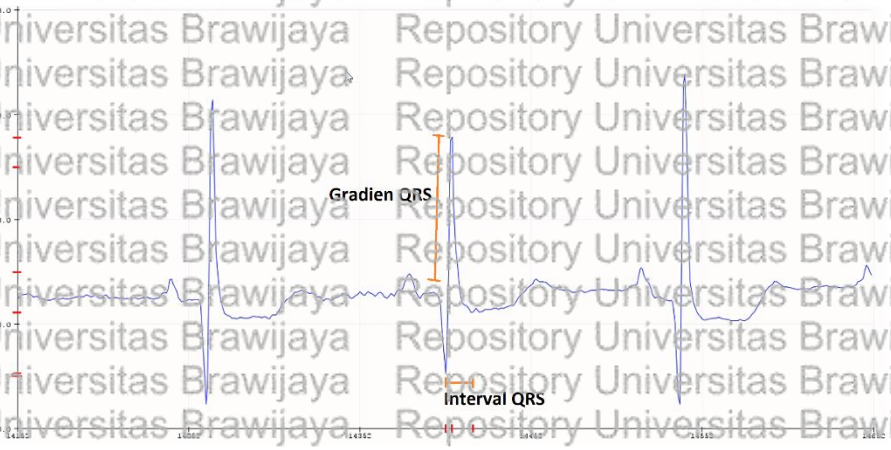
$$Gradien\ QRS = \frac{yR - yQ}{xR - xQ} \tag{2.2}$$
$$= \frac{567 - 212}{4337 - 4317}$$
$$= \frac{355}{20}$$
$$= 17.750$$

Setelah kedua nilai tersebut didapatkan, maka penentuan untuk nilai interval QRS dan gradien QRS telah selesai. Tabel 5.23 menunjukkan hasil penentuan nilai interval QRS dan gradien QRS untuk Tabel 5.21.

Tabel 5.23 Hasil Penentuan Nilai Interval QRS dan Gradien QRS

Interval QRS	Gradien QRS
62 ms	17.750

Kemudian untuk memastikan keakuratan sistem, maka akan dilakukan perhitungan untuk mendapatkan nilai interval dan gradien QRS melalui pengamatan langsung melalui *serial plotter* pada Arduino IDE. Salah satu siklus sinyal yang berhasil diakuisisi sensor AD8232 akan dijadikan bahan untuk perhitungan melalui pengamatan langsung yang ditunjukkan dalam Gambar 5.15.



Gambar 5.15 Hasil Deteksi Sinyal EKG untuk Ekstraksi Fitur

Sesuai Gambar 5 15, titik P, Q, R, S dan T dapat dikenali dengan memberikan tanda sesuai dengan karakteristiknya. Titik Q, R dan S akan digunakan untuk mendapatkan nilai fitur. Nilai dari ketiga titik tersebut berturut-turut jika diamati secara langsung terhadap sumbu y adalah 229 untuk titik Q, 568 untuk titik R dan 229 untuk titik S. Nilai tersebut merepresentasikan amplitude dari sinyal yang dideteksi. Sementara untuk nilai ketiga titik terhadap sumbu x adalah 14404 untuk titik Q, 14408 untuk titik R dan 14417 untuk titik S. Nilai tersebut merupakan *point* yang merepresentasikan waktu antara dua fungsi Serial.println() pada Arduino. Sedangkan waktu antara dua fungsi Serial.println() pada Arduino berdasarkan Tabel 5.21 adalah 5 ms. Oleh karena itu, perhitungan pada sumbu x nantinya akan dikalikan dengan 5 untuk menyamakan satuan waktu pada Arduino.

Setelah mendapatkan nilai dari masing-masing titik yang dibutuhkan, maka selanjutnya dapat dilakukan perhitungan untuk mendapatkan nilai interval dan gradien QRS berdasarkan persamaan 2.1 dan persamaan 2.2.

$$Interval\ QRS = xS - xQ \tag{2.1}$$
$$= 14417 - 14404$$
$$= 13 \times 5$$
$$= 65$$

$$Gradien\ QRS = \frac{yR - yQ}{xR - xQ} \quad (2.2)$$
$$= \frac{568 - 229}{(14408 - 14404) \times 5}$$
$$= \frac{339}{20}$$
$$= 16.95$$

Sesuai dengan perhitungan yang telah dilakukan, maka nilai interval dan gradien QRS dari siklus sinyal pada Gambar 5.15 adalah 65 dan 16.95. Jika dibandingkan dengan nilai interval dan gradien QRS yang didapatkan melalui perhitungan manual maka didapatkan selisih sebesar 3 ms untuk nilai interval QRS dan selisih sebesar 0.8 untuk nilai gradien QRS. Berdasarkan selisih tersebut dapat disimpulkan bahwa sistem mampu mendapatkan nilai interval dan gradien QRS dengan baik.

5.2.3.4 Implementasi Ekstraksi Fitur Mean, Median Interval dan Gradien QRS

Pada sub bab ini akan dijelaskan mengenai kode program untuk mendapatkan nilai rata-rata dan median dari masing-masing interval dan gradien QRS. Keempat nilai tersebut merupakan fitur yang digunakan pada sistem pendeteksi Fibrilasi Atrium. Kode program untuk ekstraksi fitur mean, median interval dan gradien QRS sendiri dapat dilihat pada Tabel 5.24

Tabel 5.24 Kode Program Ekstraksi Fitur Mean dan Median

No	*Source Code*
1	`Serial.print(F("====Memulai Ekstraksi Fitur====\n"));`
2	`Serial.print(F("Nilai Interval QRS: "));`
3	`Serial.println(intervalQRS, 3);`
4	`if(intervalQRS == 0){`
5	`Serial.print(F("Nilai Interval tidak ditemukan\n"));`
6	`}else{`
7	`buffer_mean_iQRS[periode-1] = intervalQRS;`
8	`for(int i=1; i<periode; i++){`
9	`buffer_mean_iQRS[i-1] = buffer_mean_iQRS[i];`
10	`}`
11	`}`
12	`for(int i=0; i<6; i++){`
13	`Serial.print(F("Nilai buffer mean interval QRS: "));`
14	`Serial.println(buffer_mean_iQRS[i], 3);`
15	`}`
16	
17	`data_uji.mean_iQRS = 0;`
18	`for(int i=0; i<6; i++){`
19	`data_uji.mean_iQRS += buffer_mean_iQRS[i];`
20	`buffer_med_iQRS = buffer_mean_iQRS[i];`
21	`}`
22	`data_uji.mean_iQRS /= 6;`
23	
24	`urutkanMedian(buffer_med_iQRS, periode);`
25	
26	`for(int i=0; i<6; i++){`

```
27        Serial.print(F("Nilai buffer median interval QRS: "));
28    Serial.println(buffer_med_iQRS[i], 3);
29      }
30
31      data_uji.median_iQRS = buffer_med_iQRS[2];
32      if((periode % 2)){
33        data_uji.median_iQRS += buffer_med_iQRS[3];
34      }
35      data_uji.median_iQRS /= 2;
36
37      Serial.print(F("Nilai Gradien QRS: "));
38    Serial.println(gradienQRS, 3);
39      if(gradienQRS == 0){
40        Serial.print(F("Nilai Gradien tidak ditemukan\n"));
41      }else{
42        buffer_mean_gQRS[periode-1] = gradienQRS;
43        for(int i=1; i<periode; i++){
44          buffer_mean_gQRS[i-1] = buffer_mean_gQRS[i];
45        }
46      }
47      for(int i=0; i<6; i++){
48        Serial.print(F("Nilai buffer mean gradien QRS: "));
49    Serial.println(buffer_mean_gQRS[i], 3);
50      }
51
52      data_uji.mean_gQRS = 0;
53      for(int i=0; i<6; i++){
54        data_uji.mean_gQRS += buffer_mean_gQRS[i];
55        buffer_med_gQRS[i] = buffer_mean_gQRS[i];
56      }
57      data_uji.mean_gQRS /= 6.00;
58
59      urutkanMedian(buffer_med_gQRS, periode);
60
61      for(int i=0; i<6; i++){
62        Serial.print(F("Nilai buffer median gradien QRS: "));
63    Serial.println(buffer_med_gQRS[i], 3);
64      }
65
66      data_uji.median_gQRS = buffer_med_gQRS[2];
67      if((periode % 2)){
68        data_uji.median_gQRS += buffer_med_gQRS[3];
69      }
70      data_uji.median_gQRS /= 2;
71
```

Buffer diperlukan untuk menyimpan nilai dari interval dan gradien QRS yang telah berhasil didapatkan pada sub program sebelumnya. Untuk dapat menyimpan nilai tersebut digunakan array dengan panjang data sebanyak 6 data. Hal tersebut sesuai dengan banyaknya siklus yang ditentukan untuk mengetahui subjek mengalami kondisi fibrilasi atrium atau kondisi normal.

Pada bagian kode baris 4 – 35 merupakan proses untuk menyimpan nilai interval ke dalam buffer mean interval dan buffer median interval. Untuk memasukkan nilai interval QRS, akan diperiksa terlebih dahulu apakah nilai interval QRS bernilai nol sesuai dengan baris 4. Apabila kondisi terpenuhi maka

terindikasi bahwa nilai interval belum ada. Namun apabila kondisi tidak terpenuhi, maka nilai interval QRS akan disimpan pada buffer mean interval dan memasuki perulangan untuk menyimpan nilai interval sebelumnya. Kemudian pada baris 17 – 22 merupakan proses untuk menghitung nilai rata-rata dari buffer mean interval QRS. Setiap nilai pada buffer mean akan dijumlahkan dan disimpan pada variabel mean_iQRS dan menyimpan setiap nilai buffer mean ke dalam buffer median interval. Kemudian nilai dari variabel mean_iQRS akan dibagi dengan 6. Kemudian pada baris 24 – 35 merupakan proses untuk menghitung nilai tengah dari buffer median interval QRS. Sebelum menghitung nilai tengah, akan dipanggil fungsi urutkanMedian untuk mengurutkan nilai yang masuk ke buffer median interval sesuai baris 24. Untuk mendapatkan nilai tengah maka buffer median indeks ke-2 akan ditambahkan dengan nilai buffer median indeks ke-3. Setelah dijumlahkan maka nilai tersebut akan disimpan pada variabel median_iQRS dan membagi 2 nilainya.

Pada bagian kode baris 37 – 71 merupakan proses yang sama seperti pada bagian kode baris 4 – 35. Namun terdapat perbedaan pada kedua bagian kode tersebut. Perbedaannya adalah penggunaan variabel dimana pada kode baris 4 – 35 adalah untuk menyimpan dan menentukan nilai *mean* dan *median* dari interval QRS, sementara pada bagian kode 37 – 71 adalah untuk menyimpan dan menentukan nilai *mean* dan *median* dari gradien QRS. Nilai interval dan gradien QRS yang tertampung pada *buffer* dapat dilihat pada Gambar 5.16.

```
23:37:56.584 -> ====Memulai Ekstraksi Fitur====
23:37:56.618 -> Nilai Interval QRS: 63
23:37:56.618 -> Nilai buffer mean interval QRS: 63.00
23:37:56.688 -> Nilai buffer mean interval QRS: 63.00
23:37:56.688 -> Nilai buffer mean interval QRS: 57.00
23:37:56.722 -> Nilai buffer mean interval QRS: 57.00
23:37:56.757 -> Nilai buffer mean interval QRS: 46.00
23:37:56.825 -> Nilai buffer mean interval QRS: 63.00
23:37:56.859 -> Nilai buffer median interval QRS: 46.00
23:37:56.893 -> Nilai buffer median interval QRS: 57.00
23:37:56.928 -> Nilai buffer median interval QRS: 57.00
23:37:56.996 -> Nilai buffer median interval QRS: 63.00
23:37:57.030 -> Nilai buffer median interval QRS: 63.00
23:37:57.063 -> Nilai buffer median interval QRS: 63.00
23:37:57.096 -> Nilai Gradien QRS: 11.3136
23:37:57.131 -> Nilai buffer mean gradien QRS: 9.730
23:37:57.200 -> Nilai buffer mean gradien QRS: 10.971
23:37:57.238 -> Nilai buffer mean gradien QRS: 4.834
23:37:57.266 -> Nilai buffer mean gradien QRS: 13.326
23:37:57.300 -> Nilai buffer mean gradien QRS: 17.049
23:37:57.334 -> Nilai buffer mean gradien QRS: 11.314
23:37:57.402 -> Nilai buffer median gradien QRS: 4.834
23:37:57.437 -> Nilai buffer median gradien QRS: 9.730
23:37:57.470 -> Nilai buffer median gradien QRS: 10.971
23:37:57.504 -> Nilai buffer median gradien QRS: 11.314
23:37:57.571 -> Nilai buffer median gradien QRS: 13.326
23:37:57.606 -> Nilai buffer median gradien QRS: 17.049
```

Gambar 5.16 Nilai Interval dan Gradien QRS pada Buffer

Sesuai dengan Gambar 5.16, nilai interval dan gradien QRS untuk 6 siklus dapat disimpan pada *buffer*. Kemudian untuk mendapatkan nilai *mean median* interval QRS dan *mean median* gradien QRS akan dilakukan pernitungan berdasarkan Persamaan 5.2 dan Persamaan 5.3.

a. Mean interval QRS

$$\bar{x} = \frac{x_1 + x_2 + x_3 \ldots x_n}{n}$$
$$= \frac{63 + 63 + 57 + 57 + 46 + 63}{6}$$
$$= 58{,}167$$

b. Median interval QRS

$$M_e = \frac{1}{2}\left(x_{\left(\frac{n}{2}\right)} + x_{\left(\frac{n}{2}+1\right)}\right)$$
$$= \frac{1}{2}(x_3 + x_4)$$
$$= \frac{1}{2}(57 + 63)$$
$$= 60$$

c. Mean gradien QRS

$$\bar{x} = \frac{x_1 + x_2 + x_3 \ldots x_n}{n}$$
$$= \frac{9.73 + 10.97 + 4.83 + 13.33 + 17.05 + 11.31}{6}$$
$$= 11{,}203$$

d. Median gradien QRS

$$M_e = \frac{1}{2}\left(x_{\left(\frac{n}{2}\right)} + x_{\left(\frac{n}{2}+1\right)}\right)$$
$$= \frac{1}{2}(x_3 + x_4)$$
$$= \frac{1}{2}(10.97 + 11.31)$$
$$= 11.14$$

Dari perhitungan yang dilakukan berdasarkan Gambar 5.16, didapatkan nilai fitur dari mean interval QRS = 58,167, median interval QRS = 60, mean gradien QRS = 11,203 dan median gradien QRS = 11,14.

5.2.3.5 Implementasi Pelatihan Data Latih JST

Pada sub bab ini akan dijelaskan mengenai kode program untuk melakukan pelatihan data latih JST. Hal ini diperlukan untuk menemukan bobot tetap untuk setiap fitur yang dijadikan sebagai *input* pada metode JST. Kode program pelatihan JST akan dijelaskan sesuai diagram alir pada proses perancangan sebelumnya.

Tabel 5.25 Kode Program Inisialisasi pada Pelatihan

No	*Source Code*
1	`import numpy as np`
2	`import time`
3	
4	`np.random.seed(35)`
5	
6	`data_latih = np.loadtxt("data_training_exp.txt", delimiter=',`
7	`', usecols=range(4))`
8	
9	`kelas_asal = np.loadtxt("Kelas_asal.txt")`
10	
11	`def sigFunction(x):`
12	`    return 1 / (1 + np.exp(-x))`
13	
14	`def dsigFunction(x):`
15	`    return sigFunction(x) * (1 - sigFunction (x))`
16	
17	`def softFunction(A):`
18	`    expA = np.exp(A)`
19	`    return expA / expA.sum(axis=1, keepdims=True)`
20	
21	`atr = data_fitur.shape[1]`
22	`jHidden = 5`
23	`jOutput = 2`
24	
25	`vHidden = np.random.rand(atr, jHidden)`
26	`bHidden = np.random.randn(jHidden)`
27	
28	`wOutput = np.random.rand(jHidden, jOutput)`
29	`bOutput = np.random.randn(jOutput)`
30	`lr = 10e-2`
31	
32	`array_vError = []`

Pada Tabel 5.25 berisi inisialisasi variabel yang digunakan untuk membangun model Jaringan Saraf Tiruan (JST). *Library* yang digunakan adalah *numpy* untuk perhitungan matematika dan *time* untuk mencatat waktu komputasi pelatihan sesuai pada baris 1 dan 2. Sedangkan pada baris 4 merupakan inisialisasi nilai *random* yang akan digunakan untun nilai *weight* nantinya. Kemudian pada baris 6-9 data latih yang berbentuk *text* akan disimpan pada variabel data_fitur dan kelas asal dari data latih tersebut disimpan pada variabel label_kelas. Setelah itu baris 11-19 merupakan pendefinisian fungsi aktivasi sigmoid, fungsi aktivasi *derivative* sigmoid dan fungsi aktivasi softmax.

Baris 21-23 merupakan inisialisasi dari jumlah neuron pada *input layer*, *hidden layer* dan *output layer*, dimana *hidden layer* sebanyak 2 neuron atau *node*, *input layer* sebanyak 4 *node* dan *output layer* sebanyak 2 *node*. Kemudian nilai *weight* dan *bias* untuk *hidden layer* maupun *output layer* akan diinisialisasi sesuai pada baris 25-29 dengan menggunakan nilai *random* yang diinisialisasi sebelumnya. Selanjutnya pada baris 30 terdapat nilai *learning rate* yang digunakan, yaitu $10e^{-2}$. Kemudian menyiapkan array untuk nilai *error* yang akan dihasilkan sesuai pada baris 32.

Tabel 5.26 Kode Program Fase Pelatihan

No	*Source Code*
1	`print('Training Begin...')`
2	`awal = time.time()`
3	`jPerulangan = 80000 #iterasi`
4	`for epoch in range(jPerulangan):`
5	`    ##FeedForward`
6	`    kHidden = np.dot(data_latih, wHidden) + bHidden`
7	`    aHidden = sigFunction(kHidden)`
8	
9	`    kOutput = np.dot(aHidden, wOutput) + bOutput`
10	`    aOutput = softFunction(kOutput)`
11	
12	`    ##Backpropagation`
13	
14	`    dError = aOutput - kelas_asli`
15	`    dHidden = aHidden`
16	
17	`    dwOutput = np.dot(dHidden.T, dError)`
18	
19	`    dbOutput = dError`
20	
21	`    dkOutput = wOutput`
22	`    daOutput = np.dot(dError, dkOutput.T)`
23	`    dkHidden = dsigFunction(kHidden)`
24	`    daHidden = data_latih`
25	`    dwHidden = np.dot(daHidden.T, dkHidden * daOutput)`
26	
27	`    dbHidden = daOutput * dkHidden`
28	
29	`    ## Updating Weights`
30	
31	`    wHidden -= lr * daHidden`
32	`    bHidden -= lr * dbHidden.sum(axis=0)`
33	
34	`    wOutput -= lr * dwOutput`
35	`    bOutput -= lr * dbOutput.sum(axis=0)`
36	
37	`if epoch % 10000 == 0:`
38	`        loss = np.sum(-kelas_asli * np.log(aOutput))`
39	`        print('Iterasi ke: ' + str(epoch) + '/' +`
40	`str(Perulangan) + ' Loss function value: ', loss)`
41	`        array_vError.append(loss)`

Tabel 5.26 merupakan fase yang dilakukan pada proses pelatihan JST, yaitu fase *feedforward, backpropagation* dan *updating weights*. Pada baris 1-3 merupakan fungsi untuk memulai pewaktuan dari pelatihan yang dilakukan dan menginisialisasi nilai *epocn* yang digunakan. Kemudian akan dilakukan fungsi perulangan dan memulai fase *feedforward* pada baris 4-10. Perhitungan pada fase *feedforward* dilakukan berdasarkan Persamaan 2.8 dan Persamaan 2.10. yaitu dengan melakukan perkalian nilai fitur dengan nilai *random weight* ditambah dengan nilai *bias* dan kemudian diaktivasi menggunakan fungsi aktivasi sigmoid di lapisan tersembunyi serta fungsi aktivasi softmax di lapisan keluaran.

Setelah itu program akan memasuki fase *backpropagation* sesuai baris 12-27. Pada fase *backpropagation* akan dilakukan perhitungan untuk mendapatkan nilai *weight* dan *bias* yang baru. Nilai hasil fungsi aktivasi pada fase *feedforward* akan digunakan untuk mendapatkan nilai *weight* dan *bias* tersebut. Kemudian pada baris 29-35 merupakan fase *updating weight* dimana nilai *weight* dan *bias* yang terdapat pada *hidden layer* dan *output layer* akan dikurangi dengan hasil perkalian antara nilai *learning rate* dengan nilai *weight* dan *bias* yang telah didapatkan dari fase sebelumnya. Sedangkan pada baris 37-41 merupakan pengecekan kondisi untuk *loss function cross entropy* dimana *epoch* akan berhenti ketika nilai error yang diinginkan sudah terpenuhi.

Tabel 5.27 Kode Program Fase Testing pada Pelatihan

No	*Source Code*
1	`array_Kelas = []`
2	`array_Temp = []`
3	`jTrue = 0`
4	`jFalse = 0`
5	`jKelas = np.arange(len(kelas_asli[0]))`
6	
7	`for i, item in enumerate(kOutput):`
8	`    for j, itemCols in enumerate(item):`
9	`        if all(itemCols > item[jKelas != j]):`
10	`            array_Kelas.append(j)`
11	`            if all(kelas_asli[i][j] > kelas_asli[i][jKelas`
12	`!= j]):`
13	`                array_Temp.append(1)`
14	`                jBenar += 1`
15	`            else:`
16	`                array_Temp.append(0)`
17	`                jSalah += 1`
18	
19	`akhir = time.time();`
20	`accuracy = (jumlahBenar / len(label_kelas)) * 100`
21	`print(len(label_kelas), jumlahBenar, jumlahSalah)`
22	`print('Class = ', classes)`
23	`print('Training Accuracy = ', accuracy, '%')`
24	`print('Training Finished...')`
25	`print('Training Time = ', ((akhir-awal)), 's')`
26	
27	`np.savetxt("weight_Hidden.txt", wHidden)`
28	`np.savetxt("weight_Output.txt", wOutput)`
29	`np.savetxt("bias_Hidden.txt", bHidden)`
30	`np.savetxt("bias_Output.txt", bOutput)`

Tabel 5.27 merupakan proses *testing* pada pelatihan Jaringan Saraf Tiruan. Baris 1-5 merupakan inisialisasi array dan variabel untuk menghitung jumlah benar dan salah pada pengecekan kelas dari data latih. Kemudian baris 7-17 merupakan perulangan untuk menyeleksi nilai *weight* dan *bias* dari hasil pelatihan dengan kelas asalnya. Kemudian pencatatan akhir waktu pelatihan terdapat pada baris 19. Setelah itu akan menampilkan seluruh kelas, akurasi pelatihan dan waktu pelatihan sesuai baris 21-25. Tahap terakhir adalah menyimpan nilai *weight* dan *bias* hasil pelatihan ke dalam bentuk text yang terdapat pada baris 27-30.

5.2.3.6 Implementasi Klasifikasi Metode Jaringan Saraf Tiruan

Bagian ini akan menjelaskan terkait kode program untuk proses klasifikasi menggunakan metode JST pada sistem pendeteksi Fibrilasi Atrium (FA). Kode program sendiri dapat dilihat pada Tabel 5.28.

Tabel 5.28 Kode Program Klasifikasi Jaringan Saraf Tiruan

No	*Source Code*
1	`void jstClassifier(struct JST &jst, dataUji &data_uji){`
2	`  jst.timeCalc = micros();`
3	`  //Perhitungan Lapisan Tersembunyi`
4	`  for(byte i=0; i<4; i++){`
5	`      jst.lHidden[i] = jst.wH[0][i]*data_uji.mean_iQRS;`
6	`      jst.lHidden[i] += jst.wH[1][i]*data_uji.median_iQRS;`
7	`      jst.lHidden[i]+= jst.wH[2][i]*data_uji.mean_gQRS;`
8	`      jst.lHidden[i] += jst.wH[3][i]*data_uji.median_gQRS;`
9	`      jst.lHidden[i] += jst.bH[i];`
10	`      jst.lHidden[i]= jst.sigF(jst.lHidden[i]);`
11	`  }`
12	
13	`  //Perhitungan Lapisan Keluaran`
14	`  for(byte i=0; i<2; i++){`
15	`      jst.lOutput[i] = jst.wO[0][i]*jst.lHidden[0];`
16	`      jst.lOutput[i] += jst.wO[1][i]*jst.lHidden[1];`
17	`      jst.lOutput[i] += jst.wO[2][i]*jst.lHidden[2];`
18	`      jst.lOutput[i] += jst.wO[3][i]*jst.lHidden[3];`
19	`      jst.lOutput[i] += jst.wO[4][i]*jst.lHidden[4];`
20	`      jst.lOutput[i] += jst.bO[i];`
21	`  }`
22	`  //Perhitungan softF`
23	`  float array_soft[2];`
24	`  float value_soft = 0;`
25	`  for(byte i=0; i<2; i++){`
26	`      array_soft[i] = exp(jst.lOutput[i]);`
27	`      value_soft += array_soft[i];`
28	`  }`
29	`  // hasil softF`
30	`  for(byte i=0; i<2; i++){`
31	`      jst.lOutput[i] = array_soft[i]/value_soft;`
32	`  }`
33	`  //Penentuan Kelas`
34	`  float max = 0;`
35	`  byte data_max = 0;`
36	`  float val_temp = 0;`
37	`  for(byte i=0; i<2; i++){`
38	`      if(jst.lOutput[i] > max){`
39	`          max = jst.lOutput[i];`
40	`          data_max = i;`
41	`      }`
42	`      Serial.print((String)"Robot["+i+"] = ");`
43	`      Serial.println(jst.lOutput[i], 10);`
44	`      val_temp += jst.lOutput[i];`
45	`  }`
46	`  Serial.print((String)"Jumlah weight: ");`
47	`  Serial.println(val_temp, 10);`
48	`  data_uji.kelas = data_max+'0';`
49	`  jst.fWeight = max;`

```
50    jst.timeCalc = (micros() - jst.timeCalc)/1000.f;
51    Serial.println((String)"Hasil klasifikasi adalah " +
52  jst.kelas[dataUji.kelas-'0'] + " dengan weight " + max);
53  }
```

Tabel 5.28 merupakan model arsitektur JST yang diterapkan pada Arduino IDE. Model arsitektur ini akan melakukan perhitungan pada subjek yang akan diperiksa kondisinya, apakah "FA" atau "normal". Waktu komputasi dari proses klasifikasi akan dicatat sesuai pada baris 2. Kemudian pada baris 4-11 terdapat pernitungan pada setiap *node* pada *hidden layer* dimana setiap *weignt* akan dikalikan dengan nilai fitur yang telah didapatkan dan disimpan pada array jst.lHidden[i]. Kemudian hasil array tersebut akan diaktivasi menggunakan fungsi aktivasi sigmoid sesuai baris 12.

Setelah itu program akan melakukan perhitungan untuk setiap *node* pada *output layer* sesuai pada baris 14-21. Perhitungan dilakukan dengan melakukan perkalian pada nilai *weight* dengan nilai pada array jst.lHidden[i] dan ditambah dengan nilai *bias* pada *output layer* tersebut. Hasil perhitungan tersebut akan disimpan pada jst.lOutput[i]. Selanjutnya nilai pada array jst.lOutput akan diaktivasi menggunakan fungsi aktivasi sigmoid sesuai pada baris 22-36. Perhitungan fungsi aktivasi sigmoid dilakukan dengan membagi setiap nilai pada array jst.lOutput dengan jumlah total dari nilai pada array tersebut. Kemudian akan dilaukan pencari nilai terbesar dari hasil perhitungan fungsi aktivasi softmax tersebut dan apabila nilai terbesar berhasil didapatkan maka pada serial monitor akan menampilkan nilai *weight,* waktu komputasi dan hasil klasifikasi sesuai pada baris 39-53.

BAB 6 PENGUJIAN DAN ANALISIS

Pada Bab ini menjelaskan terkait pengujian yang dilakukan pada sistem yang sudah dirancang sebelumnya dan selanjutnya akan dilakukan analisis melalui hasil yang didapatkan dari pengujian. Tujuan dilakukan pengujian terhadap sistem adalah untuk mengetahui apakah *output* yang dihasilkan oleh sistem sudah sesuai dengan yang diharapkan. Terdapat beberapa pengujian yang akan dilakukan, antara lain pengujian akurasi sensor, pengujian akurasi metode jaringan saraf tiruan dan pengujian waktu komputasi sistem.

6.1 Pengujian Pembacaan Nilai Sensor AD8232

Pada pengujian pembacaan nilai oleh sensor AD8232 adalah dengan cara melakukan perbandingan perhitungan BPM (*Beat Per Minute*) secara manual dengan perhitungan BPM melalui sensor AD8232. Hal tersebut dilakukan untuk melakukan validasi bahwa sensor telah mendeteksi sinyal EKG dengan baik.

6.1.1 Tujuan Pengujian

Pengujian pembacaan nilai oleh sensor AD8232 dilakukan dengan tujuan untuk mengetahui apakan sensor AD8232 dapat mendeteksi aktivitas listrik pada jantung dengan baik atau tidak. Dari perbandingan yang dilakukan dengan menghitung BPM melalui sensor AD8232 dan secara manual akan didapatkan nilai error sehingga tujuan tersebut telah dicapai atau tidak.

6.1.2 Prosedur Pengujian

Pengujian yang akan dilakukan tentunya memiliki prosedur yang harus dilakukan. Hal tersebut dilakukan pada pengujian ini untuk mendapatkan hasil yang sesuai dan maksimal. Prosedur pada pengujian pembacaan nilai oleh sensor AD8232 adalah:

1. Menghubungkan sistem dengan Laptop.
2. Menghubungkan kabel elektrode dengan sensor AD8232.
3. Memasang 3 buah elektrode pada tubuh subjek penelitian sesuai dengan indikator warna pada kabel, yaitu elektrode merah di dada kanan, elektrode kuning di dada kiri dan elektrode hijau di perut kanan bawah.
4. Menjalankan program sistem pada Arduino IDE.
5. Memulai rekaman layar selama 10 detik pada Serial *Plotter* untuk mendapatkan hasil sinyal EKG melalui sensor AD8232.
6. Melakukan perhitungan BPM secara manual dengan cara menekan pergelangan tangan dengan jari selama 10 detik.

Gambar 6.1 Pengujian BPM Secara Manual

7. Hasil perhitungan baik yang diperoleh melalui sensor AD8232 maupun hasil perhitungan secara manual dikalikan dengan 6 untuk mendapatkan nilai BPM.

6.1.3 Hasil Pengujian dan Analisis

Setelah mengikuti tahapan pada bagian prosedur pengujian, maka melalui *serial plotter* sinyal EKG pada subjek dapat terlihat sesuai Tabel 6.1.

Tabel 6.1 Hasil Pengujian Perhitungan BPM Melalui Sensor AD8232

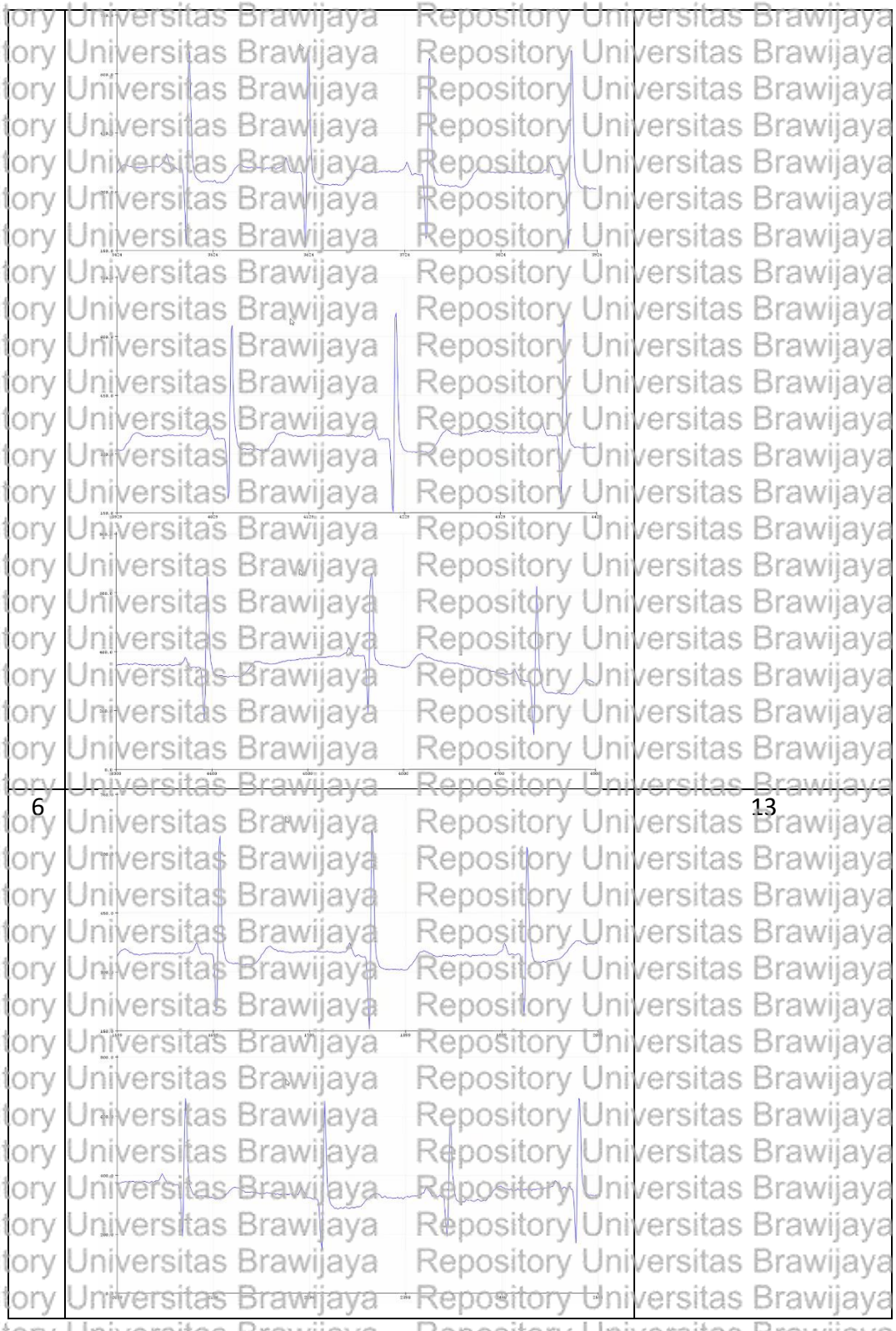
No	Tampilan Serial Plotter	Jumlah Siklus yang Terdeteksi
1		12,5

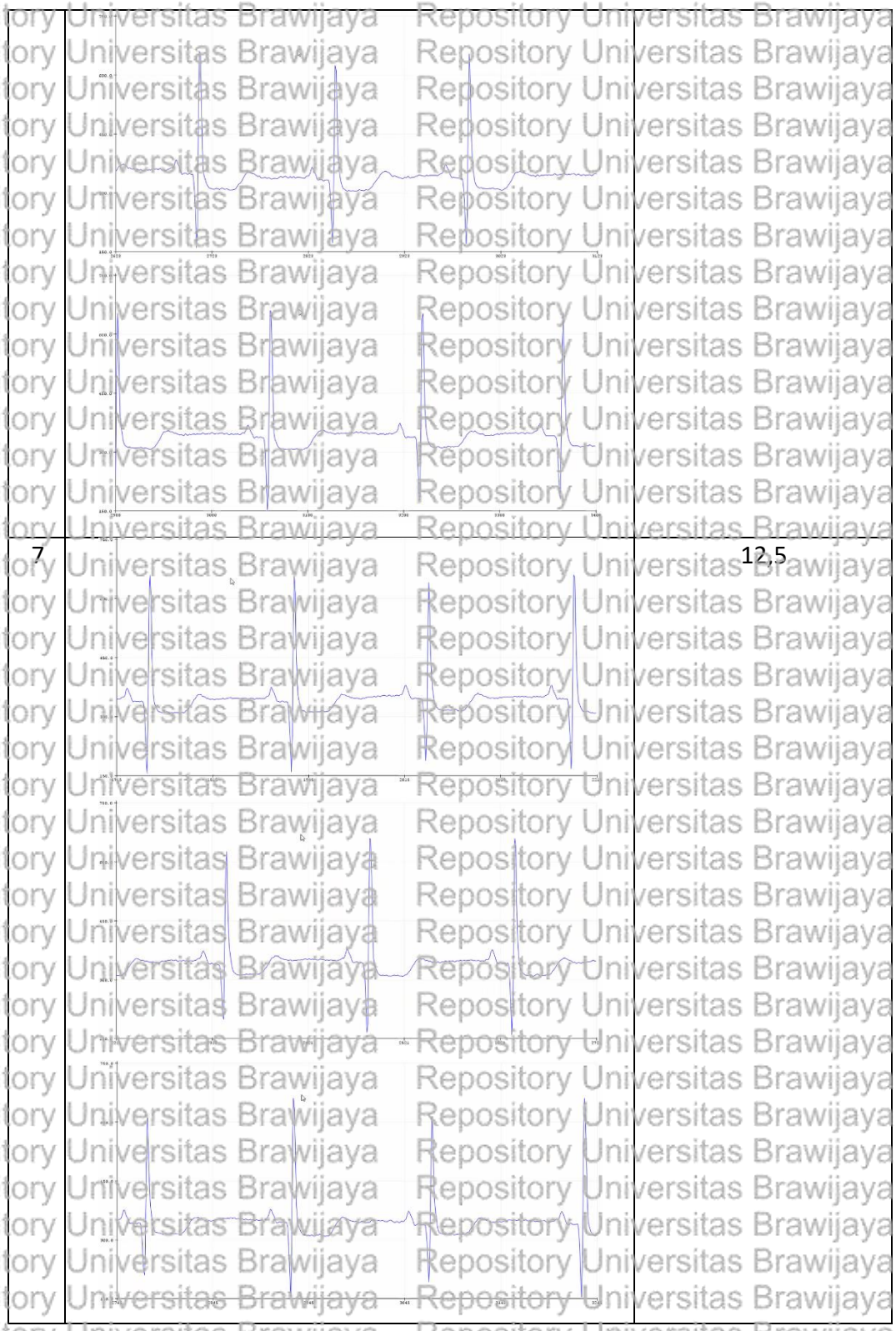
2
14

3

13

4		15,5
5		13,5


6
13


7
12,5

Repository Universitas Brawijaya
UNIVERSITAS BRAWIJAYA
REPOSITORY.UB.AC.ID

9		12,5
10		15,5

Kemudian hasil perbandingan antara perhitungan BPM yang didapatkan secara manual dan perhitungan melalui sensor AD8232 yang ditunjukkan dalam Tabel 6.2.

Tabel 6.2 Hasil Pengujian Nilai BPM

No	Nilai BPM		Selisih	Error (%)
	Manual	Sensor AD8232		
1	72	75	3	4,17%
2	78	84	6	7,69%
3	72	78	6	8,33%
4	90	93	3	4,17%
5	78	81	3	3,85%
6	84	78	6	7,14%
7	78	77	1	1,28%
8	84	78	6	7,14%

9	78	74	4	5,13%
10	90	95	5	5,55%
Rata – rata Error				5,45%

Tabel 6.2 merupakan hasil pencatatan perhitungan nilai BPM yang diambil secara manual dan melalui sensor AD8232. Setiap pengujian dilakukan selama 10 detik kemudian dikali 6. Dari setiap pengujian yang dilakukan, terdapat selisih yang merupakan nilai error (%). Nilai error tersebut dihitung untuk setiap pengujian yang dilakukan sebanyak 10 kali berdasarkan Persamaan 6.1.

$$Error\,(\%) = \frac{||BPM\ manual - BPM\ Sensor\ AD8232||}{BPM\ Manual} \times 100\% \quad (6.1)$$

$$= \frac{||72 - 75||}{72} \times 100\%$$

$$= \frac{||3||}{72} \times 100\%$$

$$= 4{,}17\%$$

Perhitungan diatas merupakan perhitungan nilai error untuk pengujian pada nomor 1. Kemudian setelah mendapatkan nilai error (%) untuk setiap pengujian, maka nilai rata-rata error (%) dari pengujian dapat diperoleh dengan perhitungan berdasarkan Persamaan 6.2.

$$Rata - rata\ error = \frac{\sum error\,(\%)}{Total\ pengujian} \quad (6.2)$$

$$= \frac{54{,}45\%}{10}$$

$$= 5{,}45\%$$

Setelah rata-rata nilai error(%) dari pengujian telah didapatkan, maka nilai akurasi dari sensor AD8232 dalam mengakuisisi sinyal EKG dapat diperoleh yaitu sebesar 95,835% dengan perhitungan berdasarkan Persamaan 6.3.

$$Akurasi = 100\% - Rata - rata\ error\,(\%) \quad (6.3)$$

$$= 100\% - 5{,}45\%$$

$$= 94{,}55\%$$

6.2 Pengujian Akurasi Metode Klasifikasi Jaringan Saraf Tiruan

Metode Jaringan Saraf Tiruan digunakan sebagai metode klasifikasi pada sistem pendeteksi fibrilasi atrium. Klasifikasi tersebut dilakukan berdasarkan empat fitur yang dipakai, antara lain rata-rata dan median dari interval QRS serta rata-rata dan median dari gradien QRS.

6.2.1 Tujuan Pengujian

Pengujian tingkat akurasi metode Jaringan Saraf Tiruan dilakukan dengan tujuan untuk mengetahui seberapa besar keakuratan sistem dalam mendeteksi penyakit Fibrilasi Atrium.

6.2.2 Prosedur Pengujian

Dalam melakukan pengujian akurasi terhadap metode Jaringan Saraf Tiruan terdapat beberapa prosedur yang harus diikuti, yaitu.

1. Menghubungkan sistem dengan Laptop.
2. Menghubungkan kabel elektrode dengan sensor AD8232.
3. Memasang 3 buah elektrode pada tubuh subjek penelitian sesuai dengan indikator warna pada kabel, yaitu elektrode merah di dada kanan, elektrode kuning di dada kiri dan elektrode hijau di perut kanan bawah.
4. Menjalankan program sistem pada Arduino IDE.
5. Membuka *serial monitor* pada Arduino IDE.

6.2.3 Hasil Pengujian dan Analisis

Masing-masing fitur yang berhasil didapatkan berperan pada metode Jaringan Saraf Tiruan sebagai masukan. Pada proses klasifikasinya akan dilakukan kalkulasi sesuai dengan perhitungan manual yang telah dijelaskan pada Bab 5. Kemudian didapatkan hasil dari pengujian tingkat akurasi metode klasifikasi Jaringan Saraf Tiruan (JST) dalam mendeteksi penyakit fibriilasi atrium ditunjukkan dalam Tabel 6.3.

Tabel 6.3 Hasil Pengujian Akurasi Metode Jaringan Saraf Tiruan

No	Fitur				Kelas Asli	Hasil Klasifikasi JST
	Interval QRS		Gradien QRS			
	Mean	Median	Mean	Median		
1	68.67	72	30.10	31.60	FA	FA
2	58.67	60	29.30	29.61	FA	Normal
3	65.33	66	10.62	10.94	FA	FA
4	60.67	62	10.64	10.51	FA	FA
5	61.33	58	12.82	12.86	FA	FA
6	59.33	58	11.80	11.88	FA	FA
7	95.17	94	22.71	16.21	FA	FA
8	70.50	70	39.91	42.32	FA	FA
9	52.67	52	10.80	11.18	FA	Normal

10	90.83	89.50	19.71	19.74	FA	FA
11	48.33	47.00	14.68	15.15	Normal	Normal
12	57.17	57.00	31.86	31.92	Normal	Normal
13	58.00	57.00	31.52	32.15	Normal	Normal
14	55.67	57.00	27.43	26.72	Normal	Normal
15	54.17	57.00	32.67	29.29	Normal	Normal
16	57.67	57.00	25.98	27.34	Normal	Normal
17	58.33	58.00	35.59	33.84	Normal	Normal
18	57.33	57.50	7.90	7.90	Normal	Normal
19	60.83	62.50	7.91	8.07	Normal	Normal
20	49.67	47.00	12.98	13.40	Normal	Normal

Sesuai dengan Tabel 6.2., hasil pengujian akurasi metode klasifikasi JST diperoleh melalui 20 data uji yang terbagi atas 10 data pasien penyakit FA dan 10 data pasien normal. Dari hasil pengujian juga ditemukan 2 data uji yang tidak sesuai dengan kelas aslinya yang ditandai dengan *cell* berwarna merah. Hal tersebut terjadi karena pada data pasien penyakit FA, nilai fitur yang didapatkan lebih mendekati kondisi pasien normal dibandingkan dengan kondisi penyakit FA. Sedangkan pada data pasien normal, yang berlaku adalah hal sebaliknya. Data latih yang dikumpulkan berpengaruh pada ketidaksesuaian yang ditemukan, dimana nilai mean median dari gradien QRS cenderung lebih rendah dan nilai mean median dari interval QRS pada kelas FA cenderung lebih tinggi dibandingkan pada kelas normal.

Untuk mendapatkan nilai akurasi dari pengujian metode klasifikasi JST pada penelitian ini dilakukan dengan menggunakan persamaan 6.4. Nilai akurasi dari metode klasifikasi JST adalah sebesar 90%.

$$Nilai\ akurasi = \frac{Total\ klasifikasi\ yang\ sesuai}{Total\ data\ uji} \times 100\% \quad (6.4)$$

$$Nilai\ akurasi = \frac{18}{20} \times 100\%$$

$$= 90\%$$

6.3 Pengujian Waktu Komputasi Metode Jaringan Saraf Tiruan

Pengujian waktu komputasi metode Jaringan Saraf Tiruan dilakukan agar sistem yang dibangun tidak hanya memilik ketepatan dalam menentukan penyakit Fibrilasi Atrium, tetapi juga memiliki kecepatan dalam menentukan penyakit tersebut. Oleh karena itu pengujian waktu komputasi metode Jaringan Saraf Tiruan diperlukan.

6.3.1 Tujuan Pengujian

Pengujian waktu komputasi metode Jaringan Saraf Tiruan dilakukan dengan tujuan untuk mengetahui waktu yang dibutuhkan oleh sistem dalam menentukan kondisi penyakit FA atau kondisi normal. Pengujian waktu komputasi sendiri akan dilakukan terhadap data yang diujikan sehingga waktu komputasi yang akan dicatat merupakan waktu komputasi testing dan satuan waktu yang digunakan pada pengujian ini adalah dalam satuan milisekon.

6.3.2 Prosedur Pengujian

Prosedur dalam melakukan pengujian waktu komputasi metode Jaringan Saraf Tiruan terdiri dari beberapa tahapan, antara lain:

1. Menghubungkan sistem dengan Laptop.
2. Menghubungkan kabel elektrode dengan sensor AD8232.
3. Memasang 3 buah elektrode pada tubuh subjek penelitian sesuai dengan indikator warna pada kabel, yaitu elektrode merah di dada kanan, elektrode kuning di dada kiri dan elektrode hijau di perut kanan bawah.
4. Menjalankan program sistem pada Arduino IDE.
5. Membuka *serial monitor* pada Arduino IDE.

6.3.3 Hasil Pengujian dan Analisis

Pengujian yang dilakukan adalah sebanyak 20 kali dan kemudian mengambil nilai rata-rata dari waktu komputasi yang dibutunkan oleh sistem. Salah satu contoh waktu komputasi pengujian ditunjukkan dalam Gambar 6.2.

```
====Memulai Klasifikasi====
Bobot[0] = 0.9999902725
Bobot[1] = 0.0000098199
Bobot total: 1.0000000000
Klasifikasi: FA
Waktu komputasi(ms): 40.00
```

Gambar 6.2 Tampilan Serial Monitor untuk Waktu Komputasi Sistem

Kemudian dari 20 pengujian yang dilakukan, hasil waktu komputasi yang dilakukan oleh sistem ditunjukkan dalam Tabel 6.4.

Tabel 6.4 Hasil Pengujian Waktu Komputasi Sistem

No	Data Uji Ke-i	Waktu Komputasi (ms)
1	1	32,08
2	2	32,09
3	3	32,10
4	4	32,09
5	5	32,09

6	6	32,08
7	7	32,10
8	8	32,09
9	9	32,08
10	10	32,10
11	11	32,08
12	12	32,09
13	13	32,10
14	14	32,10
15	15	32,08
16	16	32,10
17	17	32,09
18	18	32,08
19	19	32,08
20	20	32,08
Rata-rata waktu komputasi		32,09

Berdasarkan Tabel 6.4 waktu komputasi yang dibutuhkan oleh sistem untuk dapat melakukan klasifikasi menggunakan metode JST dalam menentukan kondisi "FA" dan kondisi "Normal" dari 20 kali pengujian didapatkan rata-rata waktu komputasi sebesar 32,09 milisekon. Waktu komputasi tersebut dimulai setelah nilai fitur yang ditentukan berhasil didapatkan dan dinormalisasi.

BAB 7 PENUTUP

Bab ini menjelaskan tentang kesimpulan yang didapatkan setelah melakukan serangkaian pengujian yang telah dilakukan pada bab sebelumnya. Kemudian dari kesimpulan yang didapatkan akan terdapat beberapa saran sehingga apabila penelitian serupa akan dilakukan dimasa mendatang maka saran tersebut dapat diimplementasikan.

7.1 Kesimpulan

Menurut rumusan masalah yang dirumuskan pada Bab Pendahuluan dan pengujian yang sudah dilakukan untuk menjawab masalah tersebut, maka didapatkan beberapa kesimpulan antara lain:

1. Pembacaan nilai yang dilakukan oleh sistem pendeteksi Fibrilasi Atrium dengan sensor AD8232 untuk dapat mendeteksi sinyal EKG dapat dilakukan dengan baik. Hal tersebut dipastikan setelah melakukan pengujian untuk menghitung nilai BPM melalui sensor AD8232 dibandingkan dengan melalui cara manual. Hasil pengujian yang didapatkan yaitu nilai akurasi sensor AD8232 pada sistem pendeteksi Fibrilasi Atrium dalam mendeteksi sinyal EKG sebesar 94,55%.
2. Metode Jaringan Saraf Tiruan yang diimplementasikan pada sistem pendeteksi Fibrilasi Atrium dapat melakukan klasifikasi terhadap data yang diujikan. Hal tersebut dipastikan setelah melakukan pengujian untuk mendapatkan nilai akurasi dari metode tersebut. Dari 20 kali pengujian terdapat 2 kali pengujian dimana metode Jaringan Saraf Tiruan salah melakukan klasifikasi karena tidak sesuai dengan kelas asalnya. Maka dari itu nilai akurasi yang didapatkan yaitu sebesar 90%.
3. Sistem pendeteksi Fibrilasi Atrium dapat melakukan klasifikasi dengan waktu yang cukup singkat. Hal terebut dipastikan setelah melakukan pengujian untuk mendapatkan waktu yang dibutuhkan sistem untuk melakukan klasifikasi terhadap data yang diujikan. Dari 20 kali pengujian waktu komputasi klasifikasi Jaringan Saraf Tiruan didapatkan hasil berupa rata-rata waktu komputasi dengan waktu selama 32,09 ms.

7.2 Saran

Dari penelitian yang telah dilakukan, terdapat beberapa saran yang mungkin dapat membantu apabila penelitian ini ingin dikembangkan lebih lanjut untuk kedepannya. Saran yang dapat dipertimbangkan dari hasil penelitian ini antara lain:

1. Peneliti selanjutnya dapat menambahkan ataupun mencoba fitur-fitur terkait karakteristik dari penyakit Fibrilasi Atrium lainnya untuk melakukan klasifikasi penyakit tersebut sehingga sistem memiliki pilihan terbaik untuk mengklasifikasikan kondisi Fibrilasi Atrium maupun kondisi Normal.

2. Peneliti selanjutnya dapat mengimplementasikan metode klasifikasi terkait pendeteksian penyakit Fibrilasi Atrium lainnya sehingga diperoleh metode klasifikasi terbaik dengan melihat tingkat akurasi yang dihasilkan oleh sistem.

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

DAFTAR REFERENSI

Adrian, K., (2018). Ciri Detak Jantung Normal dan Gangguan yang Bisa Terjadi. Available at: https://www.alodokter.com/ciri-detak-jantung-normal-dan-gangguan-yang-bisa-terjadi [Accessed 04 May 2020].

Anzihory, E., Nuryani, N. and Darmanto, D., 2016. Sistem Deteksi Fibrilasi Atrium menggunakan Fitur RR Elektokardiogram dengan Jaringan Syaraf Tiruan. *JFA (Jurnal Fisika dan Aplikasinya)*, *12*(2), pp.57-60.

Bazudewa, W.R., Satwika, I.P. and Juliharta, I.G.P.K., 2020. KLASIFIKASI ARITMIA DENGAN HEART RATE VARIABILITY ANALISIS MENGGUNAKAN METODE BACKPROPAGATION. *Jurnal Informatika dan Rekayasa Elektronik*, *3*(1), pp.1-10.

Burns, Ed., (2020). *Atrial Fibrillation*. Available at : https://litfl.com/atrial-fibrillation-ecg-library/ [Accessed 17 September 2020].

Cahya, R.A., Dewi, C. and Rahayudi, B., 2017. Klasifikasi Aritmia Dari Hasil Elektrokardiogram Menggunakan Support Vector Machine Dengan Seleksi Fitur Menggunakan Algoritma Genetika. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN*, *2548*, p.964X.

Eliana, M., Nuryani, N. and Nugroho, A.S., 2019. Deteksi Fibrilasi Atrium Menggunakan FAM yang Dikombinasikan dengan Grid Search. *Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI)*, *8*(2), pp.175-182.

Ernawati, I., Pratama, H., Y., (2017). Bagaimana cara menggunakan elektrokardiograf? Available at : https://www.dictio.id/t/bagaimana-cara-menggunakan-elektrokardiograf/6122/3 [Accessed 04 May 2020].

Fauzi, M.S.A., Rahayudi, B. and Dewi, C., 2018. Perbandingan Jaringan Saraf Tiruan LVQ Dengan Backpropagation Dalam Deteksi Dini Penyakit Jantung Koroner. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN*, *2548*, p.997X.

Fern, H.L, 2018. Aritmia (denyut jantung tidak teratur) dapat terjadi pada jantung yang sehat, namun juga dapat menjadi tanda peringatan adanya penyakit jantung. Available at: https://www.mountelizabeth.com.sg/id/healthplus/article/arrhythmia-guide [Accessed 28 April 2020]

Firdaus, I., 2019. *Indonesian Heart Association Perhimpunan Dokter Spesialis Kardiovaskular Indonesia (PERKI).* Available at. http://www.inaheart.org/news and events/news/2019/9/26/press releas e world heart day perki 2019 [Accessed 27 April 2020].

Gilang, G.A., Maulana, R. and Kurniawan, W., 2018. Implementasi Sistem Pendeteksi Premature Ventricular Contraction (Pvc) Aritmia Menggunakan Metode Naïve Bayes. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN*, *2548*, p.964X.

Gustini, E., Rahmadya, B. and Akbar, F., 2017. Sistem Deteksi Penderita Aritmania Berdasarkan Jumlah Detak Jantung Berbasis Smartphone. *Prosiding Semnastek*.

Halodoc., (2020). Fibrilasi Ventrikel. Available at : https://www.halodoc.com/kesehatan/fibrilasi-ventrikel [Accessed 04 May 2020].

Hariri, R., Hakim, L. and Lestari, R.F., 2019. Sistem Monitoring Detak Jantung Menggunakan Sensor AD8232 Berbasis Internet of Things. *InComTech*, *9*(3), pp.164-172.

Indrawati, A.N., Nugroho, A.S. and Septiyani, F., 2020, October. Detection of Obstructive Sleep Apnea Using Width and Gradient of QRS Complex With Discriminant Analysis. In *The 2nd International Seminar on Science and Technology (ISSTEC 2019)* (pp. 57-62). Atlantis Press.

Kurniawan, A., (2019). Pengertiaj Jantung Manusia, Struktur dan Fungsinya. Available at : https://www.gurupendidikan.co.id/jantung-manusia/ [Accessed 04 May 2020].

Lim, H., 2019. Waspada! Aritmia Menjadi Salah Satu Penyebab Kematian Mendadak. Available at : https://www.siloamhospitals.com/Contents/News-Event/Advertorial/2019/10/25/09/27/Waspada-Aritmia-Menjadi-Salah-Satu-Penyebab-Kematian-Mendadak [Accessed 04 May 2020]

Maknickas, V. and Maknickas, A., 2017, September. Atrial fibrillation classification using qrs complex features and lstm. In *2017 Computing in Cardiology (CinC)* (pp. 1-4). IEEE.

Mayoclinic., (2019). *Premature ventricular contractions (PVCs)*. Available at : https://www.mayoclinic.org/diseases-conditions/premature-ventricular-contractions/symptoms-causes/syc-20376757 [Accessed 04 May 2020].

Nhs., (2018). Supraventricular tachycardia (SVT). Available at : https://www.nhs.uk/conditions/supraventricular-tachycardia-svt/ [Accessed 04 May 2020].

Prasad, A.S. and Kavanashree, N., 2019, July. ECG Monitoring System Using AD8232 Sensor. In *2019 International Conference on Communication and Electronics Systems (ICCES)* (pp. 976-980). IEEE.

Rawshani, A., (2016). *Clinical ECG Interpretation*. Available at : https://ecgwaves.com/course/the-ecg-book/. [Accessed 20 Agustus 2020].

Sofiana, R., L., Maulana, R., & Utaminingrum. F., 2020. Implementasi Sistem Pendeteksi *Atrial Fibrillation* Berbasis Arduino Uno Menggunakan Metode *Support Vector Machine*. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer. e-ISSN, 2548,* 964X.

Sena, S., (2017). Pengenalan Deep Learning Part 1: Neural Network. Available at : https://medium.com/@samuelsena/pengenalan-deep-learning-8fbb7d8028ac [Accessed 04 May 2020].

Setiawan, d., (2019). Ventricular Tachycardia: Penyebab, Gejala dan, Pengobatan. Available at : https://www.honestdocs.id/ventrikular-takikardia [Accessed 04 May 2020]

Siregar, S.P. and Wanto, A., 2017. Analysis of Artificial Neural Network Accuracy Using Backpropagation Algorithm In Predicting Process (Forecasting). *International Journal Of Information System & Technology*, *1*(1), pp.34-42.

Siregar, E., Mawengkang, H., Nababan, E.B. and Wanto, A., 2019, August. Analysis of Backpropagation Method with Sigmoid Bipolar and Linear Function in Prediction of Population Growth. In *Journal of Physics: Conference Series* (Vol. 1255, No. 1, p. 012023). IOP Publishing.

Sullivan, D., (2017, September). What Are Atrial Premature Complexes? Available at : https://www.healthline.com/health/atrial-premature-complexes [Accessed 04 May 2020].

Utari, E.L., 2016. Analisa Deteksi Gelombang Qrs Untuk Menentukan Kelainan Fungsi Kerja Jantung. *Teknoin*, *22*(1).

WHO, 2017. *(World Health Organization).* [Online] Available at: https://www.who.int/en/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds) [Accessed 27 April 2020].

Willy, d., Tjin., (2018). Bradikardia. Available at : https://www.alodokter.com/bradikardia [Accessed 04 May 2020].

Yuniadi, Y., 2014. *PEDOMAN TATA LAKSANA FIBRILASI ATRIUM*. 1st ed. s.l.:Centra Communications.

Yuniadi, Y., 2017. Mengatasi Aritmia, Mencegah Kematian Mendadak. *eJournal Kedokteran Indonesia*, *5*(3), pp.46-139.

Zanury, R. (2020). Simulasi Penyakit Jantung Menggunakan Teknologi Mixed Reality.

LAMPIRAN A DATASET PENELITIAN

A.1 Data Latih Penelitian

Tabel 7.1 Data Latih Penelitian

No	Fitur				Kelas
	Interval QRS		Gradien QRS		
	Mean	Median	Mean	Median	
1	61.00	58.00	14.52	14.17	FA
2	65.33	66.00	13.45	12.17	FA
3	60.67	60.00	13.04	13.26	FA
4	60.67	62.00	23.49	23.25	FA
5	52.67	50.00	47.35	46.68	FA
6	64.00	58.00	13.50	11.64	FA
7	58.67	58.00	34.54	34.61	FA
8	105.33	102.00	20.03	20.77	FA
9	60.67	60.00	11.92	12.08	FA
10	64.00	64.00	10.68	10.56	FA
11	68.67	72.00	30.10	31.60	FA
12	80.67	80.00	29.03	28.79	FA
13	62.67	58.00	14.38	14.45	FA
14	62.00	60.00	13.67	14.38	FA
15	68.00	64.00	12.46	13.59	FA
16	58.67	56.00	23.45	22.75	FA
17	61.33	58.00	27.86	24.64	FA
18	67.33	66.00	22.93	22.33	FA
19	54.00	52.00	20.83	20.54	FA
20	63.33	64.00	31.03	32.27	FA
21	53.83	54.50	11.98	11.18	NORMAL
22	49.83	51.50	13.94	12.79	NORMAL
23	50.00	49.00	14.29	13.67	NORMAL
24	47.83	49.00	14.30	12.74	NORMAL
25	57.17	57.50	32.25	33.06	NORMAL

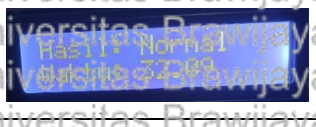
Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

26	57.50	57.50	33.12	33.18	NORMAL
27	58.17	57.00	30.60	32.69	NORMAL
28	55.50	57.00	30.60	32.69	NORMAL
29	54.83	52.50	32.19	31.22	NORMAL
30	59.67	59.50	33.15	32.91	NORMAL
31	55.00	57.00	24.40	23.83	NORMAL
32	57.83	57.00	23.75	23.48	NORMAL
33	57.00	57.50	23.59	23.92	NORMAL
34	58.83	57.50	12.34	9.08	NORMAL
35	55.83	56.00	10.22	8.80	NORMAL
36	61.67	62.50	7.44	7.41	NORMAL
37	49.50	50.00	43.33	43.03	NORMAL
38	48.33	46.50	41.74	44.31	NORMAL
39	46.33	47.00	48.86	48.89	NORMAL
40	48.50	49.50	45.84	44.26	NORMAL

A.2 Data Uji Penelitian

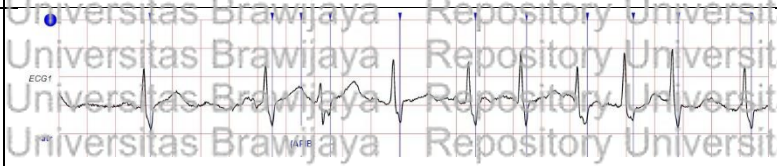
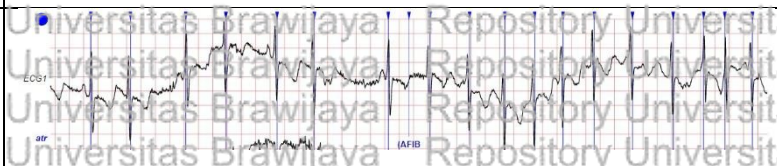
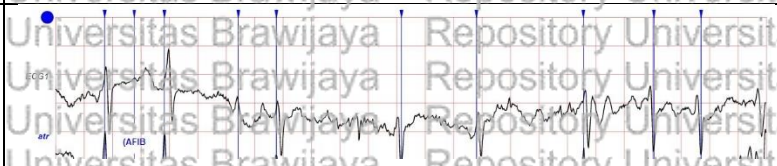
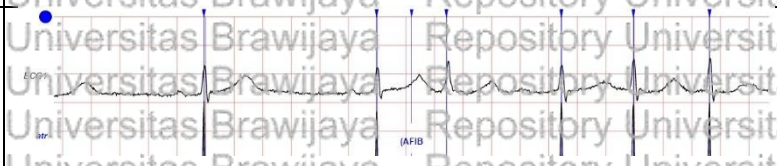
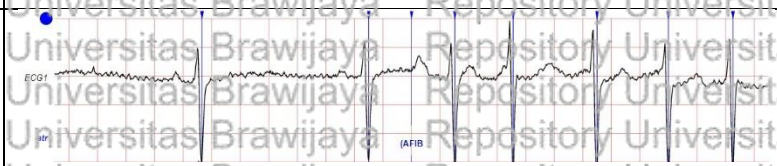
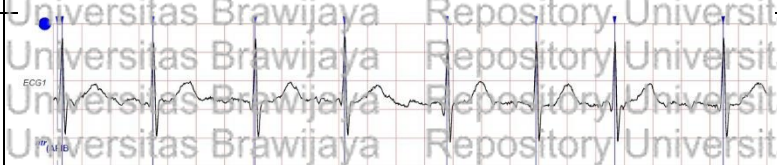
Tabel 7.2 Data Uji Penelitian

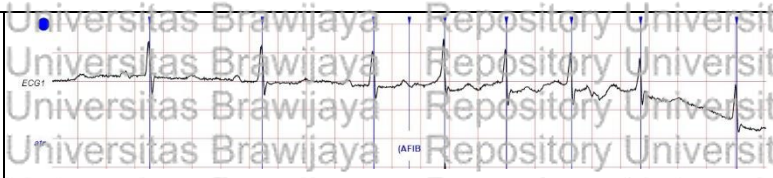
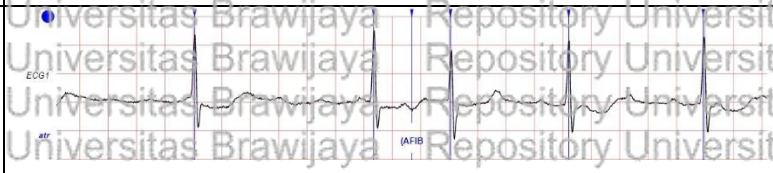
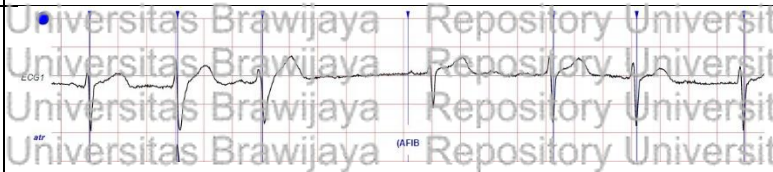
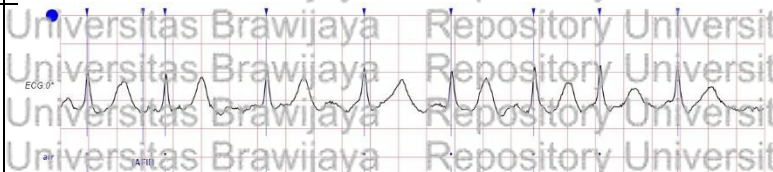
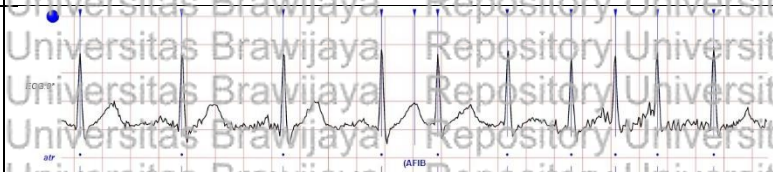
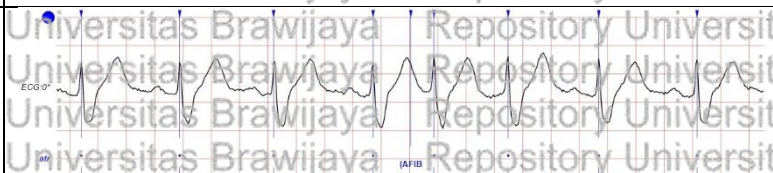
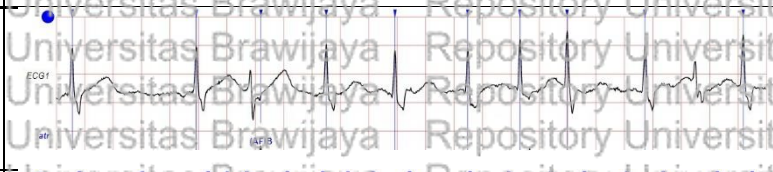
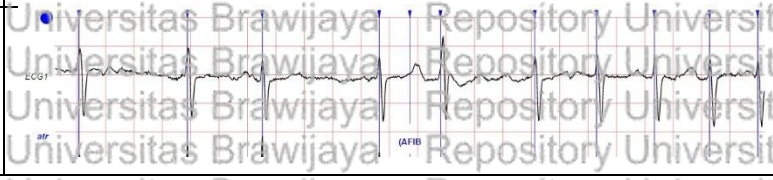
No	Fitur				Kelas Asli	Hasil Klasifikasi Yang Ditampilkan
	Interval QRS		Gradien QRS			
	Mean	Median	Mean	Median		
1	68.67	72.00	30.10	31.60	FA	
2	58.67	60.00	29.30	29.61	FA	
3	65.33	66.00	10.62	10.94	FA	
4	60.67	62.00	10.64	10.51	FA	

5	61.33	58.00	12.82	12.86	FA	
6	59.33	58.00	11.80	11.88	FA	
7	95.17	94.00	22.71	16.21	FA	
8	70.50	70.00	39.91	42.32	FA	
9	52.67	52.00	10.80	11.18	FA	
10	90.83	89.50	19.71	19.74	FA	
11	48.33	47.00	14.68	15.15	NORMAL	
12	57.17	57.00	31.86	31.92	NORMAL	
13	58.00	57.00	31.52	32.15	NORMAL	
14	55.67	57.00	27.43	26.72	NORMAL	
15	54.17	57.00	32.67	29.29	NORMAL	
16	57.67	57.00	25.98	27.34	NORMAL	

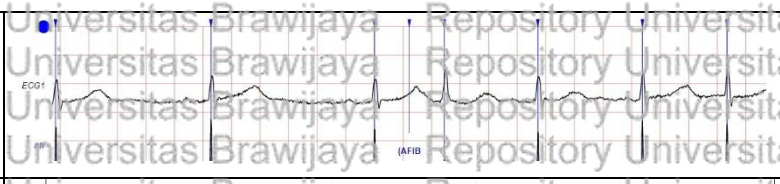
17	58.33	58.00	35.59	33.84	NORMAL	
18	57.33	57.50	7.90	7.90	NORMAL	
19	60.83	62.50	7.91	8.07	NORMAL	
20	49.67	47.00	12.98	13.40	NORMAL	

A.3 Gambar Sinyal EKG

No	Gambar	Keterangan
1		Kondisi Fibrilasi Atrium
2		Kondisi Fibrilasi Atrium
3		Kondisi Fibrilasi Atrium
4		Kondisi Fibrilasi Atrium
5		Kondisi Fibrilasi Atrium
6		Kondisi Fibrilasi Atrium

7		Kondisi Fibrilasi Atrium
8		Kondisi Fibrilasi Atrium
9		Kondisi Fibrilasi Atrium
10		Kondisi Fibrilasi Atrium
11		Kondisi Fibrilasi Atrium
12		Kondisi Fibrilasi Atrium
13		Kondisi Fibrilasi Atrium
14		Kondisi Fibrilasi Atrium
15		Kondisi Fibrilasi Atrium
16		Kondisi Fibrilasi Atrium

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

17		Kondisi Fibrilasi Atrium
18		Kondisi Normal
19		Kondisi Normal
20		Kondisi Normal

LAMPIRAN B KODE PROGRAM

B.1 Kode Program Utama

Tabel 7.3 Kode Program Pelatihan Data Latih

No	Kode Program
1	import numpy as np
2	import time
3	
4	# Pelatihan
5	
6	np.random.seed(35)
7	
8	data_fitur = np.loadtxt("data_training_exp.txt", delimiter='
9	', usecols=range(4))
10	
11	label_kelas = np.loadtxt("Label.txt")
12	
13	##fungsi aktivasi sigmoid & softmax
14	def sigmoid(x):
15	return 1 / (1 + np.exp(-x))
16	
17	def sigmoid_der(x):
18	return sigmoid(x) * (1 - sigmoid(x))
19	
20	def softmax(A):
21	expA = np.exp(A)
22	return expA / expA.sum(axis=1, keepdims=True)
23	
24	attributes = data_fitur.shape[1]
25	hidden_nodes = 5
26	output_labels = 2
27	
28	wh = np.random.rand(attributes, hidden_nodes)
29	bh = np.random.randn(hidden_nodes)
30	
31	wo = np.random.rand(hidden_nodes, output_labels)
32	bo = np.random.randn(output_labels)
33	lr = 10e-2
34	
35	error_cost = []
36	
37	print('Training Begin...')
38	awal = time.time()
39	max_iter = 80000 #iterasi
40	for epoch in range(max_iter):
41	##Fase FeedForward
42	
43	# Phase 1
44	zh = np.dot(data_fitur, wh) + bh
45	ah = sigmoid(zh)
46	
47	# Phase 2
48	zo = np.dot(ah, wo) + bo
49	ao = softmax(zo)
50	

```
    ##Fase Backpropagation

    # Phase 1

    dcost_dzo = ao - label_kelas
    dzo_dwo = ah

    dcost_wo = np.dot(dzo_dwo.T, dcost_dzo)

    dcost_bo = dcost_dzo

    # Phases 2

    dzo_dah = wo
    dcost_dah = np.dot(dcost_dzo, dzo_dah.T)
    dah_dzh = sigmoid_der(zh)
    dzh_dwh = data_fitur
    dcost_wh = np.dot(dzh_dwh.T, dah_dzh * dcost_dah)

    dcost_bh = dcost_dah * dah_dzh

    # Update Weights(Bobot)

    wh -= lr * dcost_wh
    bh -= lr * dcost_bh.sum(axis=0)

    wo -= lr * dcost_wo
    bo -= lr * dcost_bo.sum(axis=0)

    if epoch % 10000 == 0:
        loss = np.sum(-label_kelas * np.log(ao))
        print('Iter: ' + str(epoch) + '/' + str(max_iter) +
Loss function value: ', loss)
        error_cost.append(loss)
# Pengujian
akhir = time.time()
classes = []
check = []
jumlahBenar = 0
jumlahSalah = 0
_index = np.arange(len(label_kelas[0]))

for i, item in enumerate(ao):
    for j, itemCols in enumerate(item):
        if all(itemCols > item[_index != j]):
            classes.append(j)
            if all(label_kelas[i][j] > label_kelas[i][_index
!= j]):
                check.append(1)
                jumlahBenar += 1
            else:
                check.append(0)
                jumlahSalah += 1

accuracy = (jumlahBenar / len(label_kelas)) * 100
print(len(label_kelas), jumlahBenar, jumlahSalah)
print('Class = ', classes)
print('Training Accuracy = ', accuracy, '%')
print('Training Finished...')
```

```
print('Training Time = ', ((akhir-awal)*1000), 'ms')

np.savetxt("weight1_exp.txt", wh)
np.savetxt("weight2_exp.txt", wo)
np.savetxt("bias1_exp.txt", bh)
np.savetxt("bias2_exp.txt", bo)
```

Tabel 7.4 Kode Program Utama

No	Kode Program

```
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

#define PIN_LO_minus 10
#define PIN_LO_plus 11
#define PIN_output A0

int x[2][15];
int inputSignal = 0;
int Q = 0;
int R = 0;
int S = 0;
int intervalQRS = 0;
float DeltaX, DeltaY, dacConv, gradienQRS = 0;
const int periode = 7;
float buffer_mean_iQRS[periode];
float buffer_med_iQRS[periode];
float buffer_mean_gQRS[periode];
float buffer_med_gQRS[periode];
unsigned long waktu, time1;

LiquidCrystal_I2C lcd(0x27, 16, 2);
struct dataUji{
  float mean_iQRS, median_iQRS, mean_gQRS, median_gQRS;
  char kelas = 'w';
};

struct dataUji data_uji;

void normalisasi_data(dataUji &data_uji){
  data_uji.mean_iQRS = (data_uji.mean_iQRS - 46.33)/(105.33
- 46.33);
  data_uji.median_iQRS = (data_uji.median_iQRS -
46.50)/(102.00 - 46.50);
  data_uji.mean_gQRS = (data_uji.mean_gQRS - 7.44)/(48.86 -
7.44);
  data_uji.median_gQRS = (data_uji.median_gQRS -
7.41)/(48.89 - 7.41);
}

struct JST{
  float w1[4][5] = {
    {-10.2710903845, 56.1753851253, -34.8480049640,
8.8105703259, 40.6057061173},

```

```
    {-7.6716534339,    -0.4006453514,    8.4842810977,
16.3243986601, 12.4849211983},
    {-10.3430855690,    -0.3255686300,    -2.7394381673,    -
13.4890648340, -2.2909478715},
    {-20.9245183818,    28.0562463906,    39.7694681736,    -
19.0347480833, -10.0715154545}
  };

  float b1[5] = {
    6.9244434205,    -30.2436169223,    -8.8274968002,
15.7928060057, 4.0377848536
  };

  float w2[5][2] = {
    {-21.2230022363, 21.8410093066},
    {24.3024860354, -22.9735736447},
    {-24.3320534396, 25.3877183513},
    {-12.1679888766, 13.4030733071},
    {11.8356704115, -10.2378431745}
  };
  float b2[2] = {
    13.0178455700, -13.0597468791
  };

  float sigmoid(float x){
    return (float)1/(1+expf(-x));
  };

  float softmax(float *array, int i){
    static float jumlahExp = 0;
    if(jumlahExp == 0){
      for(int j = 0; j<6; j++){
        jumlahExp += expf(array[j]);
      }
    }
    return expf(array[i]) / jumlahExp;
  };

  float hidden_layer[5], output_layer[2];
  const String kelas[2] = {"FA", "Normal"};
  float waktu_komputasi, bobot_akhir;
};

struct JST jst;

void klasifikasi_jst(struct JST &jst, dataUji &data_uji){
  jst.waktu_komputasi = micros();
  // Phase 1
  // w1 = (w1*x1) + (w2*x2 + (w3*x3) + (w4*x4) + b1
  for(byte i=0; i<4; i++){
    jst.hidden_layer[i]= jst.w1[0][i]*data_uji.mean_iQRS;
    jst.hidden_layer[i]+=jst.w1[1][i]*data_uji.median_iQRS;
    jst.hidden_layer[i]+=jst.w1[2][i]*data_uji.mean_gQRS;
    jst.hidden_layer[i]+=jst.w1[3][i]*data_uji.median_gQRS;
    jst.hidden_layer[i] += jst.b1[i];
    jst.hidden_layer[i] = jst.sigmoid(jst.hidden_layer[i]);
// sigmoid
  }
```

```
107
108     // Phase 2
109     // w2 = (w1*x1) + (w2*x2) + (w3*x3) + (w4*x4) + b2
110     for(byte i=0; i<2; i++){
111       jst.output_layer[i]= jst.w2[0][i]*jst.hidden_layer[0];
112       jst.output_layer[i]+= jst.w2[1][i]*jst.hidden_layer[1];
113       jst.output_layer[i]+= jst.w2[2][i]*jst.hidden_layer[2];
114       jst.output_layer[i]+= jst.w2[3][i]*jst.hidden_layer[3];
115       jst.output_layer[i]+= jst.w2[4][i]*jst.hidden_layer[4];
116       jst.output_layer[i]+= jst.b2[i];
117     }
118
119     // exp dan jumlahnya
120     float hasil_exp[2];
121     float sigma_exp = 0;
122     for(byte i=0; i<2; i++){
123       hasil_exp[i] = exp(jst.output_layer[i]);
124       sigma_exp += hasil_exp[i];
125     }
126
127     // softmax
128     for(byte i=0; i<2; i++){
129       jst.output_layer[i]    =    hasil_exp[i]/sigma_exp;    //
130 softmax
131     }
132
133     // mencari nilai maksimum
134     float maks = 0;
135     byte indeks_maks = 0;
136     float cek_jumlah = 0;
137     for(byte i=0; i<2; i++){
138       if(jst.output_layer[i]>maks){
139         maks = jst.output_layer[i];
140         indeks_maks = i;
141       }
142       Serial.print((String)"Bobot["+i+"] = ");
143       Serial.println(jst.output_layer[i], 10);
144       cek_jumlah += jst.output_layer[i];
145     }
146     Serial.print((String)"Bobot total: ");
147     Serial.println(cek_jumlah, 10);
148     data_uji.kelas = indeks_maks+'0';
149     jst.bobot_akhir = maks;
150     jst.waktu_komputasi          =          (micros()          -
151 jst.waktu_komputasi)/1000.f;  //  Mencatat  waktu  komputasi
152 dari awal hingga berakhirnya proses klasifikasi JST
153     //Serial.println((String)"Hasil  klasifikasi  adalah  " +
154 jst.kelas[dataUji.kelas-'0'] + " dengan bobot " + maks);
155 }
156
157 void urutkanMedian(float *dataArray, byte jumlahData){
158   byte i = 0;
159   for(byte i=0; i<jumlahData; i++){
160     byte k;
161     for(byte j=1; j<jumlahData; j++){
162       if(dataArray[j-1] > dataArray[j]){
163         float temp = dataArray[j-1];
164         dataArray[j-1] = dataArray[j];
165         dataArray[j] = temp;
```

```
    }
  }
 }
}
void setup() {
  Serial.begin(9600);
  pinMode(PIN_LO_minus, INPUT);
  pinMode(PIN_LO_plus, INPUT);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("    DETEKSI");
  lcd.setCursor(0,1);
  lcd.print("          FA");
  lcd.clear();
}
void loop() {
  // put your main code here, to run repeatedly:
  static JST jst;
  static dataUji data_uji;
  inputSignal = analogRead(PIN_output);
  waktu = millis();
  if((analogRead(PIN_LO_minus) ==
1)||(analogRead(PIN_LO_plus) == 1)){
    Serial.print('!');
  }else{
    Serial.println(inputSignal);
  }

  for(int i=14; i>=0; i--){
    if(i == 0){
      x[0][i] = inputSignal;
      x[1][1] = waktu;
      continue;
    }
    x[0][i] = x[0][i-1];
    x[1][i] = x[1][i-1];
  }

  if ((x[0][1]>420 && x[0][1] > x[0][0] && x[0][1]>x[0][2])
|| R > 0){
    R++;
    if (x[0][0] > x[0][1] && x[0][1] < x[0][R]){
      S++;
    }
  }

  if (S>0){
    for(int i = R; i<15; i++){
      if(x[0][i] >= x[0][i-1] && x[0][i-1] < x[0][i-2]){
        Q = i-1;
      }
    }
    intervalQRS = x[1][S] - x[1][Q];
    DeltaY = x[0][R] - x[0][Q];
    DeltaX = x[1][R] - x[1][Q];
    dacConv = ((DeltaY*3300)/1024);
```

```
    gradienQRS = dacConv/DeltaX;

    //time1 = millis();
    Serial.print(F("====Memulai Ekstraksi Fitur====\n"));
    Serial.print(F("Nilai Interval QRS: "));
Serial.println(intervalQRS, 3);
    if(intervalQRS == 0){
      Serial.print(F("Nilai Interval tidak ditemukan\n"));
    }else{
      buffer_mean_iQRS[periode-1] = intervalQRS;
      for(int i=1; i<periode; i++){
        buffer_mean_iQRS[i-1] = buffer_mean_iQRS[i];
      }
    }
    for(int i=0; i<6; i++){
      Serial.print(F("Nilai buffer mean interval QRS: "));
Serial.println(buffer_mean_iQRS[i], 3);
    }

    data_uji.mean_iQRS = 0;
    for(int i=0; i<6; i++){
      data_uji.mean_iQRS += buffer_mean_iQRS[i];
      buffer_med_iQRS[i] = buffer_mean_iQRS[i];
    }
    data_uji.mean_iQRS /= 6;

    urutkanMedian(buffer_med_iQRS, periode);

    for(int i=0; i<6; i++){
      Serial.print(F("Nilai buffer median interval QRS: "));
Serial.println(buffer_med_iQRS[i], 3);
    }

    data_uji.median_iQRS = buffer_med_iQRS[2];
    if((periode % 2)){
      data_uji.median_iQRS += buffer_med_iQRS[3];
    }
    data_uji.median_iQRS /= 2;

    Serial.print(F("Nilai Gradien QRS: "));
Serial.println(gradienQRS, 3);
    if(gradienQRS == 0){
      Serial.print(F("Nilai Gradien tidak ditemukan\n"));
    }else{
      buffer_mean_gQRS[periode-1] = gradienQRS;
      for(int i=1; i<periode; i++){
        buffer_mean_gQRS[i-1] = buffer_mean_gQRS[i];
      }
    }
    for(int i=0; i<6; i++){
      Serial.print(F("Nilai buffer mean gradien QRS: "));
Serial.println(buffer_mean_gQRS[i], 3);
    }

    data_uji.mean_gQRS = 0;
    for(int i=0; i<6; i++){
      data_uji.mean_gQRS += buffer_mean_gQRS[i];
      buffer_med_gQRS[i] = buffer_mean_gQRS[i];
    }
```

```
        data_uji.mean_gQRS /= 6.00;
    urutkanMedian(buffer_med_gQRS, periode);

    for(int i=0; i<6; i++){
      Serial.print(F("Nilai buffer median gradien QRS: "));
Serial.println(buffer_med_gQRS[i], 3);
    }

    data_uji.median_gQRS = buffer_med_gQRS[2];
    if((periode % 2)){
      data_uji.median_gQRS += buffer_med_gQRS[3];
    }
    data_uji.median_gQRS /= 2;

    if(buffer_mean_iQRS[0] != 0 && buffer_med_iQRS[0] != 0
&& buffer_mean_gQRS[0] != 0 && buffer_med_gQRS[0] != 0){
      Serial.print(F("Nilai mean interval QRS: "));
Serial.println(data_uji.mean_iQRS, 3);
      Serial.print(F("Nilai median interval QRS: "));
Serial.println(data_uji.median_iQRS, 3);
      Serial.print(F("Nilai mean gradien QRS: "));
Serial.println(data_uji.mean_gQRS, 3);
      Serial.print(F("Nilai median gradien QRS: "));
Serial.println(data_uji.median_gQRS, 3);
      normalisasi_data(data_uji);
      Serial.print(F("====Memulai Klasifikasi====\n"));
      klasifikasi_jst(jst, data_uji);
      Serial.print(F("Klasifikasi: "));
Serial.println(jst.kelas[data_uji.kelas-'0']);
      Serial.print(F("Waktu komputasi(ms): "));
Serial.println(jst.waktu_komputasi);
      Serial.print(F("Bobot: "));
Serial.println(jst.bobot_akhir, 4);
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Hasil: ");
lcd.print(jst.kelas[data_uji.kelas-'0']);
      lcd.setCursor(0,1);
      lcd.print("Waktu: "); lcd.print(jst.waktu_komputasi);
      memset(buffer_med_gQRS, 0, sizeof(buffer_med_gQRS));
      memset(buffer_mean_gQRS, 0,
sizeof(buffer_mean_gQRS));
      memset(buffer_med_iQRS, 0, sizeof(buffer_med_iQRS));
      memset(buffer_mean_iQRS, 0,
sizeof(buffer_mean_iQRS));
    }
    intervalQRS = 0; gradienQRS = 0; DeltaY = 0; DeltaX = 0;
    Q = 0; R = 0; S = 0; dacConv = 0;
  }
}
```