PERANCANGAN PROGRAM DAN ANALISIS OPTIMALISASI DISTRIBUSI KANAL RADIO SIARAN FM BERBASIS JAVA UNTUK PROPINSI DAERAH ISTIMEWA YOGYAKARTA

SKRIPSI

OLEH

HARYANTO SURYALI

04 04 03 044 X

DEPARTEMEN TEKNIK ELEKTRO

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

GENAP 2007/2008

PERANCANGAN PROGRAM DAN ANALISIS OPTIMALISASI DISTRIBUSI KANAL RADIO SIARAN FM BERBASIS JAVA UNTUK PROPINSI DAERAH ISTIMEWA YOGYAKARTA

SKRIPSI

Oleh

HARYANTO SURYALI

04 04 03 044X

SKRIPSI INI DIAJUKAN UNTUK MELENGKAPI SEBAGIAN PERSYARATAN MENJADI SARJANA TEKNIK

DEPARTEMEN TEKNIK ELEKTRO

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

GENAP 2007/2008

PERNYATAAN KEASLIAN SKRIPS

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul:

PERANCANGAN PROGRAM DAN ANALISIS OPTIMALISASI DISTRIBUSI KANAL RADIO SIARAN FM BERBASIS JAVA UNTUK PROPINSI DAERAH ISTIMEWA YOGYAKARTA

yang dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada program studi Teknik Elektro Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia, sejauh yang saya ketahui bukan merupakan tiruan atau duplikasi dari skripsi yang sudah diduplikasikan dan atau pernah dipakai untuk mendapatkan gelar kesarjanaan di lingkungan Universitas Indonesia maupun di Perguruan Tinggi atau Instansi manapun, kecuali bagian yang sumber informasinya dicantumkan sebagaimana mestinya.

Depok, 23 April 2008

Haryanto Suryali

NPM 04 04 03 044X

PENGESAHAN

Skripsi dengan judul :

PERANCANGAN PROGRAM DAN ANALISIS OPTIMALISASI DISTRIBUSI KANAL RADIO SIARAN FM BERBASIS JAVA UNTUK PROPINSI DAERAH ISTIMEWA YOGYAKARTA

Dibuat untuk melengkapi sebagian persyaratan menjadi sarjana teknik pada program studi Teknik Elektro Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia dan disetujui untuk diajukan dalam sidang ujian skripsi.

Depok, 23 April 2008

Dosen Pembimbing

Prof. Dr. Ir. Dadang Gunawan, M.Eng

NIP 131 475 421

UCAPAN TERIMA KASIH

Puji dan syukur kepada Allah Tritunggal atas anugerahNya dan rahmat-Nya sehingga skripsi ini dapat diselesaikan dengan baik. Penulis juga mengucapkan terima kasih kepada:

Prof. Dr. Ir. Dadang Gunawan, M.Eng

Selaku dosen pembimbing yang telah bersedia meluangkan waktu untuk memberikan pengarahan, diskusi dan bimbingan serta persetujuan sehingga skripsi ini dapat selesai dengan baik.

Haryanto Suryali
NPM 04 04 03 044X
Departemen Teknik Elektro

Dosen Pembimbing
Prof. Dr. Ir. Dadang Gunawan, M.Eng

PERANCANGAN PROGRAM DAN ANALISIS OPTIMALISASI DISTRIBUSI KANAL RADIO SIARAN FM BERBASIS JAVA UNTUK PROPINSI DAERAH ISTIMEWA YOGYAKARTA

ABSTRAK

Distribusi kanal radio siaran FM yang selama ini tidak seimbang dan bijaksana menyebabkan penggunaan sumber daya kanal tidak optimal. Daerah-daerah berkembang mengalami kesulitan menambah porsi kanal dari yang telah ditetapkan. Distribusi kanal radio siaran FM untuk tiap-tiap propinsi ditetapkan pada Keputusan Menteri no 15 tahun 2003.

Sebuah program sederhana dirancang untuk mencari keberadaan nomor-nomor kanal yang masih mungkin dialokasikan bagi suatu daerah. Nomor-nomor kanal ini merupakan nomor-nomor kanal yang bebas interferensi, baik terhadap semua pemancar-pemancar potensi interferensi, maupun terhadap pemancar itu sendiri.

Propinsi yang dioptimalisasi dan dianalisa adalah propinsi Daerah Istimewa Yogyakarta. Penentuan daerah-daerah potensi interferensi dilakukan secara manual dengan menggunakan *software* Chirplus BC, yaitu daerah-daerah pada propinsi JawaTengah.

Hasil simulasi dan analisa memperlihatkan bahwa lima pemancar dari total delapan pemancar yang dimiliki oleh propinsi tersebut memiliki nomor-nomor kanal yang mengalami interferensi dengan pemancar lain. Terdapat lima pemancar yang kepadanya masih dapat dialokasikan nomor-nomor kanal baru, dengan syarat nomor-nomor kanal yang mengalami interferensi diabaikan.

Kata Kunci : Java, kanal FM

Haryanto Suryali
NPM 04 04 03 044X
Electrical Engineering Department

Counsellor
Prof. Dr. Ir. Dadang Gunawan, M.Eng

JAVA-BASED SOFTWARE DESIGN AND ANALYSIS OPTIMALIZATION CHANNEL DISTRIBUTION OF FM BROADCASTING RADIO AND ANALYSIS THE OUTPUT OF THE PROGRAM FOR THE PROVINCE OF YOGYAKARTA

ABSTRACT

The distribution of FM broadcasting radio channels that has been allocated is both unbalance and unwise. This results in a less optimum use of channel source. Growing regions are also in trouble of adding new channels beside what has been allocated for them before. The distribution of FM broadcasting radio channels is stated on Keputusan Menteri no 15 tahun 2003.

A simple program is built to search the existence of new channels that are possible to be allocated for a transmitter in a region. These channels are interference-free channels, from both other potential-interfering transmitters, and the transmitter itself.

The province selected to be optimized and examined is the province of Yogyakarta. The selection of potential-interfering regions is done manually, by means of software Chirplus BC. This selection focuses on regions whose boundary located next to the province of D.I Yogyakarta, that is Central Java. The frequency spacing between two transmitters is calculated based on the value of protection ratio that has been calculated from data-processing.

The analysis shows that five among total eight transmitters have channels that conduct interference with another transmitter. If the channels that conduct interference are neglected, then there are still five transmitters that have possibilities to be allocated to new channels.

Keywords : Java, FM channels

DAFTAR ISI

Halaman

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

I.1. LATAR BELAKANG

Diawali pada tahun 1970 ketika pemerintah memperbolehkan Radio Siaran Non Pemerintah atau Radio Swasta dengan dikeluarkannya Peraturan Pemerintah Nomor 55 Tahun 1970 tentang Radio Siaran Non Pemerintah. Pada waktu itu, dua instansi pemerintah ditunjuk sebagai regulator teknis untuk mengatur radio siaran, yaitu [2]:

1. Ditjen Postel-Dephub/Depparpostel, untuk mengatur frekuensi Radio Siaran Non Pemerintah
2. Ditjen RTF-Deppen, untuk mengatur frekuensi RRI.

Dalam perjalanannya, koordinasi kedua instansi ini kurang baik sehingga menimbulkan banyak kesulitan di masa yang akan datang.

Pada tahun 1971 dikeluarkan S.K. Menhub No.25/T/1971 yang menyatakan bahwa alokasi frekuensi FM adalah 100 – 108 MHz. Pada waktu itu teknologi yang digunakan masih *monophonic*, dimana transmisi sinyal hanya menggunakan satu kanal, dengan daya pancar maskimum 25 Watt [2].

Teknologi *stereophonic* yang menggunakan beberapa kanal sekaligus untuk mentransmisi sinyal dimulai pada tahun 1982 berdasarkan Kep. Menhub No.KM.262/PT.307/Phb-82, dengan bandwidth 250 kHz dan daya pancar maksimum 100 Watt. Berdasarkan Keputusan Menteri tahun 1982 (KM 15) ini juga ditetapkan spasi kanal sebesar 350 kHz dari 100.2 sampai 107.8 MHz, yaitu sebanyak 22 kanal [2].

Berdasarkan Keputusan Menparpostel No.KM73/PT.102/MPPT/94, alokasi frekuensi radio FM yang digunakan adalah 87 sampai 108 MHz [2]. Alokasi frekuensi ini diperluas dari yang sebelumnya, yaitu 100 sampai 108 MHz, dengan sehingga dapat memenuhi permintaan siaran radio FM yang semakin banyak.

Pada rentang tahun 1998 – 2001, Ditjen RTF-Deppen bubar sehingga pemberian izin menjadi sepenuhnya dikerjakan oleh Ditjen Postel. Akan tetapi pada prakteknya, pada tahun 2001 – 2003, ada regulator pemerintah daerah juga memberikan izin siaran selain Ditjen Postel. Hal ini mengakibatkan koordinasi yang kurang baik dan menyebabkan kekacauan penataan frekuensi radio FM.

Pemberian izin siaran selama ini dilakukan dengan "siapa cepat dia yang dapat" dan kurang memiliki perencanaan yang matang sehingga terjadi pemborosan dan ketidakefisienan dalam penggunaan sumber daya frekuensi. Perencanaan yang matang seharusnya memperhitungkan luas daerah cakupan, pemakaian ulang frekuensi (*frequency reuse*) dan pengkanalan yang baik sehingga penggunaan sumber frekuensi menjadi optimal.

Hal ini berakibat pada ketidakmerataan jumlah kanal yang mencolok antara kota-kota besar dan kota-kota kecil yang berada dalam wilayah yang berdekatan. Pemberian izin yang kurang bijaksana ini membuat jumlah kanal diboroskan pada kota-kota besar, sehingga tidak ada, atau tinggal sangat sedikit kanal yang masih bisa diberikan pada kota-kota kecil. Padahal setiap daerah berpotensi untuk berkembang menjadi kota yang lebih besar. Hal ini menjadi masalah jangka panjang. Dengan sumber daya kanal terbatas, maka pemerintah tidak akan sanggup lagi memenuhi permintaan kanal FM.

Pada bulen Mei sampai Juni tahun 2002, Ditjen Postel bekerja sama dengan tenaga ahli dari ITU, Mr. S. Razavi, melakukan penelitian dan pembenahan kembali perencanaan frekuensi FM [2]. Hasilnya adalah bahwa perencanaan yang selama ini dilakukan kurang tepat. Salah satunya adalah Keputusan Menhub tahun 1982 yang menetapkan penggunaan spasi kanal sebesar 350 kHz. Spasi ini menyalahi rekomendasi ITU-R dan menyulitkan masyarakat karena tidak semua pesawat radio memiliki kemampuan untuk menangkap frekuensi sampai ketelitian 50Khz.

Oleh karena itu diputuskan untuk membuat rencana induk penataan frekuensi dimana distribusi kanal dilakukan dengan berdasar pada wilayah layanan. Spasi antar kanal menggunakan 100KHz, sehingga terdapat 204 kombinasi kanal dari frekuensi 87.6 – 107.9 MHz. Tiga kanal 107.7, 107.8, dan 107.9 MHz diperuntukkan bagi radio

komunitas yang memiliki daya pancar rendah dan jangkauan terbatas (maksimal 2.5 km) [2].

Berdasarkan Keputusan Mentri no. 15, telah disusun daftar penomoran kanal radio siaran FM untuk setiap kota dalam setiap propinsi di Indonesia. Penomoran kanal tersebut tidak boleh mengalami interferensi satu dengan yang lainnya. Semakin dekat jarak antara dua kota, maka semakin besar spasi frekuensi yang dibutuhkan sehingga tidak terjadi interferensi. Hal ini berarti perencanaan penomoran kanal harus dilakukan dengan seksama sehingga menjamin semua nomor-nomor kanal tersebut tidak ada yang mengalami interferensi.

Penomoran kanal bukan hanya harus memperhatikan faktor bebas interferensi, tetapi juga harus disusun sedemikian rupa sehingga sumber daya frekuensi yang ada digunakan secara maksimal. Hal ini dilakukan dengan *frequency reuse*, yaitu kondisi dimana jarak antara dua kota cukup jauh sehingga mereka dapat menggunakan frekuensi yang sama, atau dengan kata lain mereka dapat menggunakan nomor kanal yang sama. Oleh karena itu, adalah hal yang sangat penting untuk menghitung spasi frekuensi minimal antara masing-masing kota, sehingga pada kota-kota tersebut dapat diberlakukan *co-channel*. Dengan cara ini sumber daya frekuensi yang ada dapat digunakan secara optimal.

Seperti yang telah dinyatakan sebelumnya, pengkanalan radio siaran FM di Indonesia sangat tidak merata dan tidak bijaksana. Kota-kota besar mendapatkan porsi jumlah kanal yang jauh lebih banyak dibanding dengan kota-kota kecil disekitarnya. Akibat yang terjadi adalah menipisnya kemungkinan bagi kota-kota kecil dan berkembang untuk mendapatkan alokasi jumlah kanal baru.

Melalui skripsi ini, sebuah *software* sederhana dibuat untuk melakukan optimalisasi kanal radio siaran FM. Adapaun tugas dari program sederhana ini adalah untuk medeteksi apakah masih ada sejumlah nomor kanal dari nomor-nomor yang tersedia (kanal 1 sampai kanal 201, karrena nomor kanal dari 202-204 tidak diperuntukkan bagi komersil) yang masih bisa dialokasikan untuk kota yang ditunjuk. Artinya, kota tersebut dapat menambah porsi jumlah nomor kanalnya dengan nomor

yang dihasilkan oleh program. Nomor kanal tersebut dijamin bebas interferensi, baik dengan pemancar lain, ataupun pemancar itu sendiri.

JAVA digunakan sebagai bahasa pemrograman dalam merancang *software* ini. Hal ini dikarnakan JAVA memiliki beberapa keuntungan seperti *platform independence*, berbasis *Object Oriented Programming*, dan merupakan *free ware*. Hal ini akan dijelaskan kemudian pada Bab II.

Diharapkan dengan hadirnya program sederhana ini, analisa terhadap penomoran kanal dalam KM15 dan optimalisasi pengkanalan untuk tiap daerah dapat dikerjakan dengan lebih cepat dan akurat.

I.2. TUJUAN PENULISAN

Skripsi ini dibuat untuk melakukan optimalisasi distribusi kanal radio siaran FM, yaitu untuk mencari keberadaan nomor-nomor kanal yang masih dapat dialokasikan bagi suatu pemancar.

I.3. BATASAN MASALAH

Masalah yang akan diangkat dalam seminar ini dibatasi hanya pada penataan frekuensi radio FM, dan daerah yang akan dianalisa oleh penulis dibatasi pada propinsi Daerah Istimewa Yogyakarta.

I.4. SISTEMATIKA PENULISAN

Makalah seminar ini dibagi menjadi 7 bab, dengan perincian sebagai berikut. Bab satu adalah pendahuluan yang berisi sejarah dan latar belakang permasalahan yang muncul pada perencanaan frekuensi FM, serta solusi yang ditawarkan. Bab dua berisi dasar teori yang berhubungan dengan penataan frekuensi radio FM, bab tiga berisi dasar teori singkat mengenai bahasa pemrograman JAVA. Bab empat berisi perancangan *software* optimalisasi kanal radio siaran FM, bab lima berisi perhitungan

spasi frekuensi yang menjadi *input* bagi program. Bab enam berisi analisa output program terhadap propinsi Daerah Istimewa Yogyakarta. Bab tujuh adalah kesimpulan.

BAB II
PARAMETER PENATAAN FREKUENSI DALAM OPTIMALISASI DISTRIBUSI KANAL RADIO SIARAN FM

2.1. *PROTECTION RATIO*

Protection ratio merupakan selisih kuat medan minimum antara dua pancaran (yang diinginkan dan yang tidak diinginkan) pada suatu titik batas sehingga tidak terjadi interferensi satu dengan yang lainnya. Nilai *protection ratio* merupakan nilai yang harus dipenuhi sehingga dapat memberikan kualitas layanan yang baik.

Nilai *protection ratio* diberikan untuk keadaan *steady* dan *tropospheric*. Keadaan *steady* adalah keadaan dimana pengukuran *protection ratio* dilakukan dalam rentang waktu yang lama secara terus menerus. Sedangkan kondisi *tropospheric* adalah kondisi dimana pengukuran *protection ratio* dilakukan pada rentang waktu yang sangat singkat karena adanya suatu interferensi singkat yang menggangu dan bersifat anomali.

Gambar 2.1 *Protection ratio*

Maka *protection ratio* dapat diformulasikan dengan [2]:

Protection ratio (PR) = E *wanted* – E *unwanted* 2-1

Ada dua jenis layanan penerimaan dalam radio siaran FM yang juga menentukan karakteristik *protection ratio*: penerimaan *monophonic* dan *stereophonic*. Penerimaan *monophonic* merupakan metode penerimaan FM yang pertama dan hanya menggunakan kanal tunggal dalam memancarkan gelombang untuk menghasilkan suara. *Stereophonic* merupakan metode dimana sinyal ditransmisikan melalui dua atau lebih kanal yang berbeda melalui proses yang disebut *multiplexing*. Kekurangan metode *stereophonic* adalah jangkauan pancaran yang mampu dicapai hanyalah setengah dari pancaran *monophonic* pada tingkat daya yang sama.

Frekuensi deviasi merupakan penyimpangan frekuensi yang terjadi, dan dibagi menjadi dua, yaitu frekuensi deviasi ±75 kHz, dan ±50 kHz [3]. Tabel 2.1 dan 2.2 memperlihatkan nilai *protection ratio* pada sistem dengan frekuensi deviasi ±75 kHz dan ±50 kHz, pada penerimaan *monophonc* dan *stereophonic* untuk berbagai spasi frekuensi.

Tabel 2.1 *Protection ratio* dengan Frekuensi Deviasi ±75 kHz [3]

Carrier frequency spacing (kHz)	***Protection ratio***			
	Monophonic		***Stereophonic***	
	Steady interference	***Tropospheric interference***	***Steady interference***	***Tropospheric interference***
000	36.0	28.0	45.0	37.0
100	12.0	12.0	33.0	25.0
200	6.0	6.0	7.0	7.0
300	-7.0	-7.0	–7.0	–7.0
400	–20.0	–20.0	–20.0	–20.0

Tabel 2.2 *Protection ratio* dengan Frekuensi Deviasi ±50 kHz [3]

Carrier frequency spacing (kHz)	*Protection ratio*			
	Monophonic		*Stereophonic*	
	Steady interference	*Tropospheric interference*	*Steady interference*	*Tropospheric interference*
0	39.0	32.0	49.0	41.0
100	12.0	12.0	33.0	25.0
200	–2.5	–2.5	7.0	7.0
300	–10.0	–10.0	–7.0	–7.0
400	–20.0	–20.0	–20.0	–20.0

2.2. *MINIMUM FIELD STRENGTH*

Kuat medan minimum (E_{min}) adalah pancaran kuat medan yang harus dipenuhi oleh sebuah pemancar untuk mencukupi wilayah layanannya. Satuan untuk kuat medan adalah dBµV/m.

Pada kondisi dimana terdapat interferensi dari peralatan industri, maka besar kuat medan untuk menghasilkan layanan yang memuaskan tidak boleh lebih kecil dari nilai yang diperlihatkan pada tabel 2.3.

Tabel 2.3 Minimum *Field Strength* dengan Adanya Interferensi dari Peralatan Industri [3]

	Layanan	
Kelas Area	***Monophonic* dB (µV/m)**	***Stereophonic* dB (µV/m)**
Metropolitan	70	74
Urban	60	66
Rural	48	54

Pada kondisi dimana tidak terdapat interferensi dari peralatan industri, maka besar kuat medan ditunjukkan pada Tabel 2.4 dapat digunakan.

Tabel 2.4 Minimum *Field Strength* tanpa Interferensi Peralatan Industri [3]

Layanan	
Monophonic dB (µV/m)	**Stereophonic dB (µV/m)**
34	48

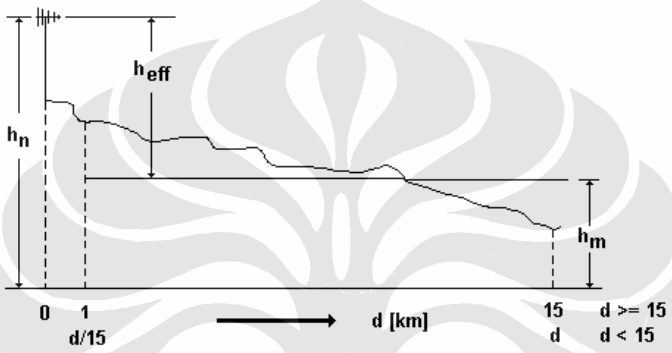
Tabel 2.5 memperlihatkan daftar radius wilayah layanan yang menjadi kebijakan pemerintah Indonesia bagi radio siaran FM berdasarkan jenis kelasnya [2]:

Tabel 2.5 Radius Jangkauan Siaran Radio FM [6]

Jenis kelas	Radius
Kelas A (metropolitan)	30 km
Kelas B (urban)	20 km
Kelas C (rural)	12 km
Kelas D (radio komunitas)	2.5 km

2.3. EHAAT

EHAAT merupakan singkatan dari *Effective High Above Average Terrain*. EHAAT adalah ketinggian yang diukur dari rata-rata permukaan tanah pada rentang 1 sampai 15 km dari tempat awal pengukuran menuju arah titik akhir. Gambar 2.2 memperlihatkan pengukuruan EHAAT.



Gambar 2.2 Pengukuran EHAAT [4]

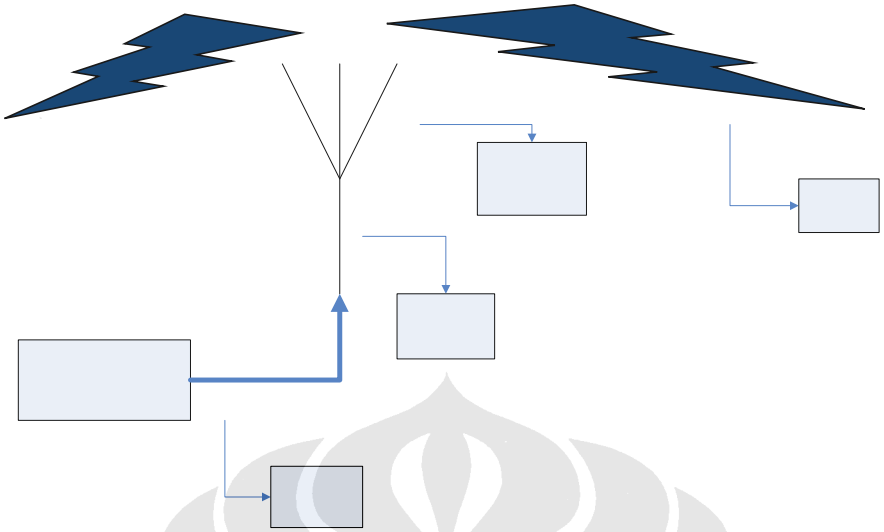
Tinggi rata-rata permukaan tanah dihitung dengan mencari nilai rata-rata dari 141 titik ketinggian (h_i) pada jarak sampai dengan 15 km dari titik asal. Persamaan 2-2 merumuskan perhitungan EHAAT [4]:

$$h_m = \frac{\sum_{i=0}^{140} h_i}{141} \qquad 2\text{-}2$$

Tedapat dua jenis tinggi antena efektif yang digunakan: tinggi antena efektif untuk pemancar dan tinggi antena efektif untuk penerima.

2.4. ERP (*EFFECTIVE RADIATIF POWER*)

ERP merupakan daya yang dipancarkan antena setelah dikurangi *loss feeder*. ERP dinyatakan dalam satuan dBkW. Gambar 2.3 memperlihatkan perhitungan ERP.



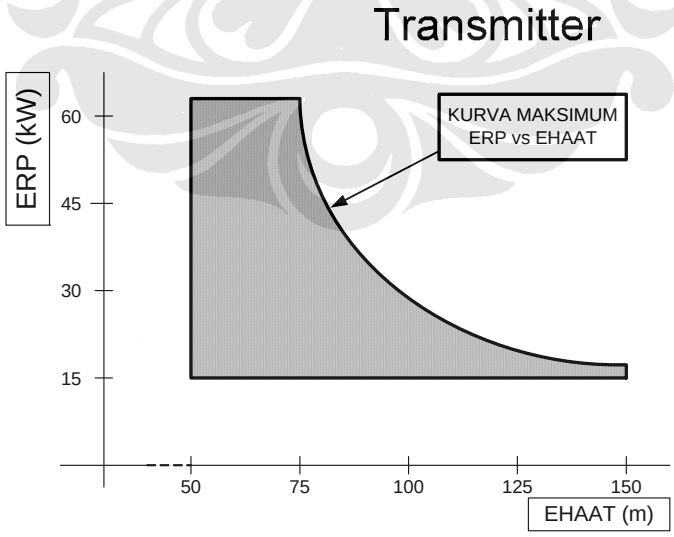
Gambar 2.3 Perhitungan ERP

$$ERP\ (dBkW) = P_{tx} - L_{feed} + G_{antenna} \qquad 2\text{-}3$$

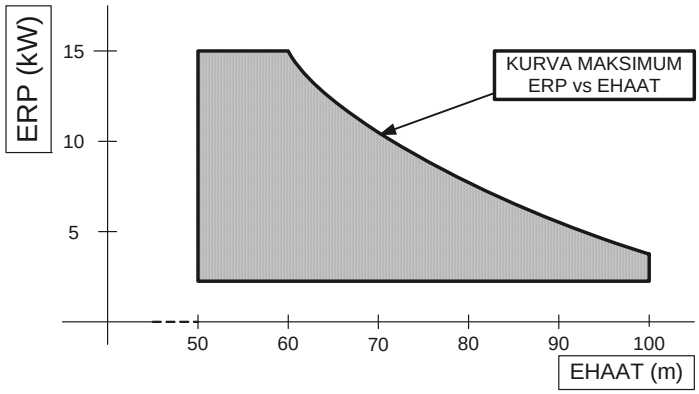
dengan

$$dBkW = 10\ Log\ (kW)$$

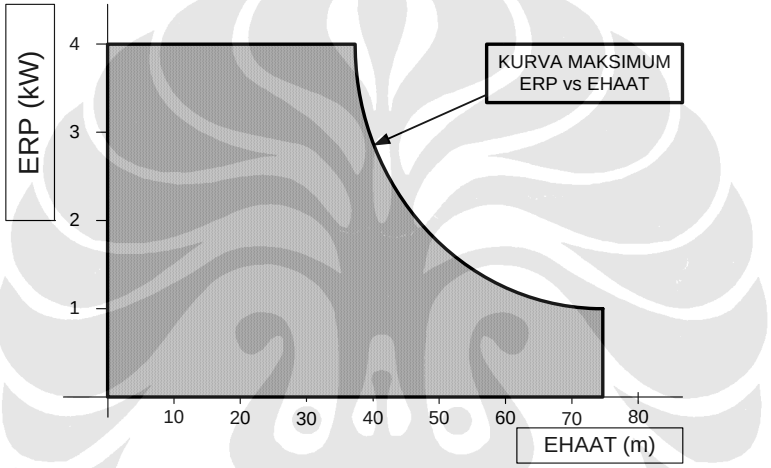
Gambar 2.4 (a) sampai dengan (d) memperlihatkan grafik hubungan antara EHAAT dengan ERP yang diambil dari KM 15:



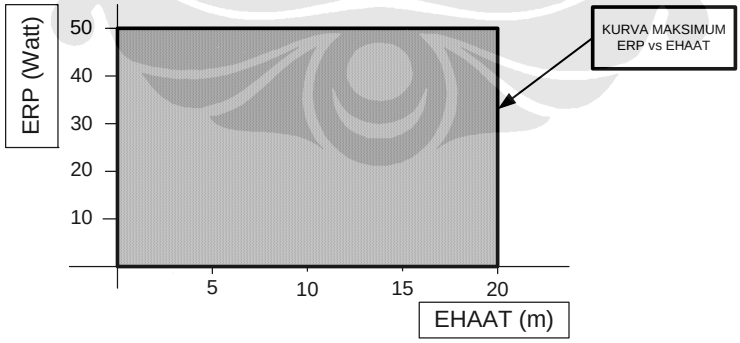
Gambar 2.4(a) Hubungan antara EHAAT dan ERP untuk Kelas A [6]



Gambar 2.4(b) Hubungan antara EHAAT dan ERP untuk Kelas B [6]



Gambar 2.4(c) Hubungan antara EHAAT dan ERP untuk Kelas C [6]



Gambar 2.4(d) Hubungan antara EHAAT dan ERP untuk Kelas D [6]

2.5. INTERFERENSI *STEADY* DAN *TROPOSPHERIC*

Seperti pada *protection ratio*, interferensi dapat dibedakan menjadi interferensi *steady* atau interferensi *tropospheric*. Maka adalah hal yang penting untuk menentukan apakah suatu interferensi digolongkan *steady* atau *tropo*. Kriteria untuk melakukan hal ini diambil dari konsep *nusance field*. *Nuisance field* adalah kuat medan pemancar yang menginterferensi pada ERP tertentu ditambah dengan *protection ratio* yang relevan[1].

Nuisance field untuk interferensi *steady* dirumuskan sebagai berikut [3]:

$$E_s = P + E(50,50) + A_s \qquad 2\text{-}4$$

Sedangkan untuk interferensi *tropospheric*:

$$E_t = P + E(50,T) + A_t \qquad 2\text{-}5$$

Dengan

P : ERP. (dB(1 kW)) dari pemancar yeng menginterferensi

A : *protection ratio* (dB)

$E(50, T)$: kuat medan (dB(μV/m)) pemancar yang menginterferensi dan ditambahkan selama T% dari total waktu. Besar T adalah 1%, yang merupakan hasil konferensi VHF/FM di Geneva tahun 1984.

Suatu interferensi disebut steady apabila hasil *nuisance field steady* lebih besar dari *nuisance field tropo*, dan sebaliknya. Penentuan jenis gangguan ini adalah penting sehingga dapat ditentukan nilai *protection ratio* mana yang harus digunakan.

2.6. *SERVICE AREA* DAN *COVERAGE AREA*

Service Area adalah wilayah cakupan layanan oleh sebuah *transmitter* tertentu dimana didalam wilayah tersebut sinyal yang dikirim dijamin dapat diterima dengan baik. Nilai *minimum field strength* pada ujung batas *service area* harus melebihi *nuisance field*.

Sedangkan *coverage area* adalah wilayah cakupan yang dapat dijangkau oleh sebuah *transmitter* dimana sinyal masih dapat diterima dengan baik, tetapi tanpa

[1] Diterjemahkan dari Annex 1, rekomendasi ITU-R BS.412-9

BAB I
PENDAHULUAN

I.1. LATAR BELAKANG

Diawali pada tahun 1970 ketika pemerintah memperbolehkan Radio Siaran Non Pemerintah atau Radio Swasta dengan dikeluarkannya Peraturan Pemerintah Nomor 55 Tahun 1970 tentang Radio Siaran Non Pemerintah. Pada waktu itu, dua instansi pemerintah ditunjuk sebagai regulator teknis untuk mengatur radio siaran, yaitu [2]:

1. Ditjen Postel-Dephub/Depparpostel, untuk mengatur frekuensi Radio Siaran Non Pemerintah
2. Ditjen RTF-Deppen, untuk mengatur frekuensi RRI.

Dalam perjalanannya, koordinasi kedua instansi ini kurang baik sehingga menimbulkan banyak kesulitan di masa yang akan datang.

Pada tahun 1971 dikeluarkan S.K. Menhub No.25/T/1971 yang menyatakan bahwa alokasi frekuensi FM adalah 100 – 108 MHz. Pada waktu itu teknologi yang digunakan masih *monophonic*, dimana transmisi sinyal hanya menggunakan satu kanal, dengan daya pancar maskimum 25 Watt [2].

Teknologi *stereophonic* yang menggunakan beberapa kanal sekaligus untuk mentransmisi sinyal dimulai pada tahun 1982 berdasarkan Kep. Menhub No.KM.262/PT.307/Phb-82, dengan bandwidth 250 kHz dan daya pancar maksimum 100 Watt. Berdasarkan Keputusan Menteri tahun 1982 (KM 15) ini juga ditetapkan spasi kanal sebesar 350 kHz dari 100.2 sampai 107.8 MHz, yaitu sebanyak 22 kanal [2].

Berdasarkan Keputusan Menparpostel No.KM73/PT.102/MPPT/94, alokasi frekuensi radio FM yang digunakan adalah 87 sampai 108 MHz [2]. Alokasi frekuensi ini diperluas dari yang sebelumnya, yaitu 100 sampai 108 MHz, dengan sehingga dapat memenuhi permintaan siaran radio FM yang semakin banyak.

Pada rentang tahun 1998 – 2001, Ditjen RTF-Deppen bubar sehingga pemberian izin menjadi sepenuhnya dikerjakan oleh Ditjen Postel. Akan tetapi pada prakteknya, pada tahun 2001 – 2003, ada regulator pemerintah daerah juga memberikan izin siaran selain Ditjen Postel.

Hal ini mengakibatkan koordinasi yang kurang baik dan menyebabkan kekacauan penataan frekuensi radio FM.

Pemberian izin siaran selama ini dilakukan dengan "siapa cepat dia yang dapat" dan kurang memiliki perencanaan yang matang sehingga terjadi pemborosan dan ketidakefisienan dalam penggunaan sumber daya frekuensi. Perencanaan yang matang seharusnya memperhitungkan luas daerah cakupan, pemakaian ulang frekuensi (*frequency reuse*) dan pengkanalan yang baik sehingga penggunaan sumber frekuensi menjadi optimal.

Hal ini berakibat pada ketidakmerataan jumlah kanal yang mencolok antara kota-kota besar dan kota-kota kecil yang berada dalam wilayah yang berdekatan. Pemberian izin yang kurang bijaksana ini membuat jumlah kanal diboroskan pada kota-kota besar, sehingga tidak ada, atau tinggal sangat sedikit kanal yang masih bisa diberikan pada kota-kota kecil. Padahal setiap daerah berpotensi untuk berkembang menjadi kota yang lebih besar. Hal ini menjadi masalah jangka panjang. Dengan sumber daya kanal terbatas, maka pemerintah tidak akan sanggup lagi memenuhi permintaan kanal FM.

Pada bulen Mei sampai Juni tahun 2002, Ditjen Postel bekerja sama dengan tenaga ahli dari ITU, Mr. S. Razavi, melakukan penelitian dan pembenahan kembali perencanaan frekuensi FM [2]. Hasilnya adalah bahwa perencanaan yang selama ini dilakukan kurang tepat. Salah satunya adalah Keputusan Menhub tahun 1982 yang menetapkan penggunaan spasi kanal sebesar 350 kHz. Spasi ini menyalahi rekomendasi ITU-R dan menyulitkan masyarakat karena tidak semua pesawat radio memiliki kemampuan untuk menangkap frekuensi sampai ketelitian 50Khz.

Oleh karena itu diputuskan untuk membuat rencana induk penataan frekuensi dimana distribusi kanal dilakukan dengan berdasar pada wilayah layanan. Spasi antar kanal menggunakan 100KHz, sehingga terdapat 204 kombinasi kanal dari frekuensi 87.6 – 107.9 MHz. Tiga kanal 107.7, 107.8, dan 107.9 MHz diperuntukkan bagi radio komunitas yang memiliki daya pancar rendah dan jangkauan terbatas (maksimal 2.5 km) [2].

Berdasarkan Keputusan Mentri no. 15, telah disusun daftar penomoran kanal radio siaran FM untuk setiap kota dalam setiap propinsi di Indonesia. Penomoran kanal tersebut tidak boleh mengalami interferensi satu dengan yang lainnya. Semakin dekat jarak antara dua kota, maka semakin besar spasi frekuensi yang dibutuhkan sehingga tidak terjadi interferensi. Hal ini berarti perencanaan penomoran kanal harus dilakukan dengan seksama sehingga menjamin semua nomor-nomor kanal tersebut tidak ada yang mengalami interferensi.

Penomoran kanal bukan hanya harus memperhatikan faktor bebas interferensi, tetapi juga harus disusun sedemikian rupa sehingga sumber daya frekuensi yang ada digunakan secara maksimal. Hal ini dilakukan dengan *frequency reuse*, yaitu kondisi dimana jarak antara dua kota cukup jauh sehingga mereka dapat menggunakan frekuensi yang sama, atau dengan kata lain mereka dapat menggunakan nomor kanal yang sama. Oleh karena itu, adalah hal yang sangat penting untuk menghitung spasi frekuensi minimal antara masing-masing kota, sehingga pada kota-kota tersebut dapat diberlakukan *co-channel*. Dengan cara ini sumber daya frekuensi yang ada dapat digunakan secara optimal.

Seperti yang telah dinyatakan sebelumnya, pengkanalan radio siaran FM di Indonesia sangat tidak merata dan tidak bijaksana. Kota-kota besar mendapatkan porsi jumlah kanal yang jauh lebih banyak dibanding dengan kota-kota kecil disekitarnya. Akibat yang terjadi adalah menipisnya kemungkinan bagi kota-kota kecil dan berkembang untuk mendapatkan alokasi jumlah kanal baru.

Melalui skripsi ini, sebuah *software* sederhana dibuat untuk melakukan optimalisasi kanal radio siaran FM. Adapaun tugas dari program sederhana ini adalah untuk medeteksi apakah masih ada sejumlah nomor kanal dari nomor-nomor yang tersedia (kanal 1 sampai kanal 201, karrena nomor kanal dari 202-204 tidak diperuntukkan bagi komersil) yang masih bisa dialokasikan untuk kota yang ditunjuk. Artinya, kota tersebut dapat menambah porsi jumlah nomor kanalnya dengan nomor yang dihasilkan oleh program. Nomor kanal tersebut dijamin bebas interferensi, baik dengan pemancar lain, ataupun pemancar itu sendiri.

JAVA digunakan sebagai bahasa pemrograman dalam merancang *software* ini. Hal ini dikarnakan JAVA memiliki beberapa keuntungan seperti *platform independence*, berbasis *Object Oriented Programming*, dan merupakan *free ware*. Hal ini akan dijelaskan kemudian pada Bab II.

Diharapkan dengan hadirnya program sederhana ini, analisa terhadap penomoran kanal dalam KM15 dan optimalisasi pengkanalan untuk tiap daerah dapat dikerjakan dengan lebih cepat dan akurat.

I.2. TUJUAN PENULISAN

Skripsi ini dibuat untuk melakukan optimalisasi distribusi kanal radio siaran FM, yaitu untuk mencari keberadaan nomor-nomor kanal yang masih dapat dialokasikan bagi suatu pemancar.

I.3. BATASAN MASALAH

Masalah yang akan diangkat dalam seminar ini dibatasi hanya pada penataan frekuensi radio FM, dan daerah yang akan dianalisa oleh penulis dibatasi pada propinsi Daerah Istimewa Yogyakarta.

I.4. SISTEMATIKA PENULISAN

Makalah seminar ini dibagi menjadi 7 bab, dengan perincian sebagai berikut. Bab satu adalah pendahuluan yang berisi sejarah dan latar belakang permasalahan yang muncul pada perencanaan frekuensi FM, serta solusi yang ditawarkan. Bab dua berisi dasar teori yang berhubungan dengan penataan frekuensi radio FM, bab tiga berisi dasar teori singkat mengenai bahasa pemrograman JAVA. Bab empat berisi perancangan *software* optimalisasi kanal radio siaran FM, bab lima berisi perhitungan spasi frekuensi yang menjadi *input* bagi program. Bab enam berisi analisa output program terhadap propinsi Daerah Istimewa Yogyakarta. Bab tujuh adalah kesimpulan.

BAB II
PARAMETER PENATAAN FREKUENSI DALAM OPTIMALISASI DISTRIBUSI KANAL RADIO SIARAN FM

2.1. *PROTECTION RATIO*

Protection ratio merupakan selisih kuat medan minimum antara dua pancaran (yang diinginkan dan yang tidak diinginkan) pada suatu titik batas sehingga tidak terjadi interferensi satu dengan yang lainnya. Nilai *protection ratio* merupakan nilai yang harus dipenuhi sehingga dapat memberikan kualitas layanan yang baik.

Nilai *protection ratio* diberikan untuk keadaan *steady* dan *tropospheric*. Keadaan *steady* adalah keadaan dimana pengukuran *protection ratio* dilakukan dalam rentang waktu yang lama secara terus menerus. Sedangkan kondisi *tropospheric* adalah kondisi dimana pengukuran *protection ratio* dilakukan pada rentang waktu yang sangat singkat karena adanya suatu interferensi singkat yang menggangu dan bersifat anomali.

Gambar 2.1 *Protection ratio*

Maka *protection ratio* dapat diformulasikan dengan [2]:

Protection ratio (PR) = E *wanted* – E *unwanted* 2-1

Ada dua jenis layanan penerimaan dalam radio siaran FM yang juga menentukan karakteristik *protection ratio*: penerimaan *monophonic* dan *stereophonic*. Penerimaan *monophonic* merupakan metode penerimaan FM yang pertama dan hanya menggunakan kanal tunggal dalam memancarkan gelombang untuk menghasilkan suara. *Stereophonic* merupakan metode dimana sinyal ditransmisikan melalui dua atau lebih kanal yang berbeda melalui proses yang disebut *multiplexing*. Kekurangan metode *stereophonic* adalah jangkauan pancaran yang mampu dicapai hanyalah setengah dari pancaran *monophonic* pada tingkat daya yang sama.

Frekuensi deviasi merupakan penyimpangan frekuensi yang terjadi, dan dibagi menjadi dua, yaitu frekuensi deviasi ±75 kHz, dan ±50 kHz [3]. Tabel 2.1 dan 2.2 memperlihatkan nilai *protection ratio* pada sistem dengan frekuensi deviasi ±75 kHz dan ±50 kHz, pada penerimaan *monophonc* dan *stereophonic* untuk berbagai spasi frekuensi.

Tabel 2.1 *Protection ratio* dengan Frekuensi Deviasi ±75 kHz [3]

Carrier frequency spacing (kHz)	***Protection ratio***			
	Monophonic		***Stereophonic***	
	Steady interference	***Tropospheric interference***	***Steady interference***	***Tropospheric interference***
000	36.0	28.0	45.0	37.0
100	12.0	12.0	33.0	25.0
200	6.0	6.0	7.0	7.0
300	-7.0	-7.0	–7.0	–7.0
400	–20.0	–20.0	–20.0	–20.0

Tabel 2.2 *Protection ratio* dengan Frekuensi Deviasi ±50 kHz [3]

Carrier frequency spacing (kHz)	*Protection ratio*			
	Monophonic		*Stereophonic*	
	Steady interference	*Tropospheric interference*	*Steady interference*	*Tropospheric interference*
0	39.0	32.0	49.0	41.0
100	12.0	12.0	33.0	25.0
200	–2.5	–2.5	7.0	7.0
300	–10.0	–10.0	–7.0	–7.0
400	–20.0	–20.0	–20.0	–20.0

2.2. *MINIMUM FIELD STRENGTH*

Kuat medan minimum (E_{min}) adalah pancaran kuat medan yang harus dipenuhi oleh sebuah pemancar untuk mencukupi wilayah layanannya. Satuan untuk kuat medan adalah dBµV/m.

Pada kondisi dimana terdapat interferensi dari peralatan industri, maka besar kuat medan untuk menghasilkan layanan yang memuaskan tidak boleh lebih kecil dari nilai yang diperlihatkan pada tabel 2.3.

Tabel 2.3 Minimum *Field Strength* dengan Adanya Interferensi dari Peralatan Industri [3]

Kelas Area	Layanan	
	***Monophonic* dB (µV/m)**	***Stereophonic* dB (µV/m)**
Metropolitan	70	74
Urban	60	66
Rural	48	54

Pada kondisi dimana tidak terdapat interferensi dari peralatan industri, maka besar kuat medan ditunjukkan pada Tabel 2.4 dapat digunakan.

Tabel 2.4 Minimum *Field Strength* tanpa Interferensi Peralatan Industri [3]

Layanan	
Monophonic dB (µV/m)	**Stereophonic dB (µV/m)**
34	48

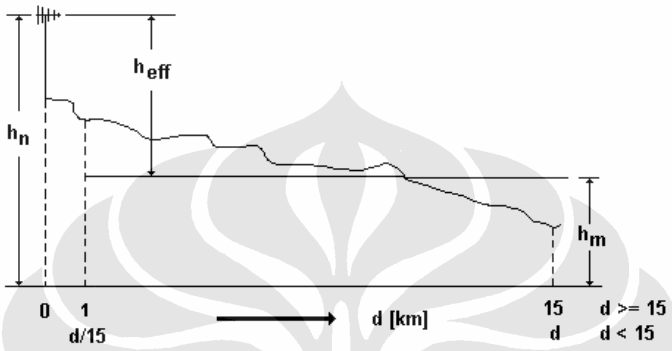
Tabel 2.5 memperlihatkan daftar radius wilayah layanan yang menjadi kebijakan pemerintah Indonesia bagi radio siaran FM berdasarkan jenis kelasnya [2]:

Tabel 2.5 Radius Jangkauan Siaran Radio FM [6]

Jenis kelas	Radius
Kelas A (metropolitan)	30 km
Kelas B (urban)	20 km
Kelas C (rural)	12 km
Kelas D (radio komunitas)	2.5 km

2.3. EHAAT

EHAAT merupakan singkatan dari *Effective High Above Average Terrain*. EHAAT adalah ketinggian yang diukur dari rata-rata permukaan tanah pada rentang 1 sampai 15 km dari tempat awal pengukuran menuju arah titik akhir. Gambar 2.2 memperlihatkan pengukuruan EHAAT.



Gambar 2.2 Pengukuran EHAAT [4]

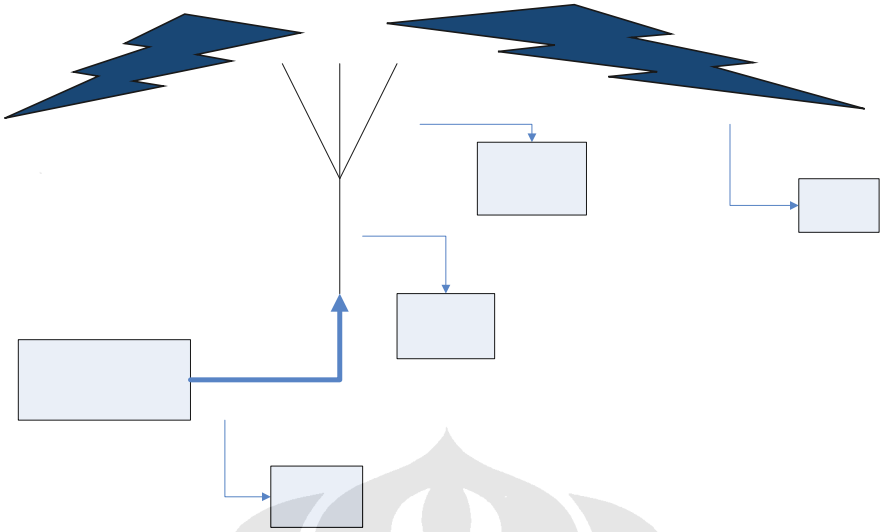
Tinggi rata-rata permukaan tanah dihitung dengan mencari nilai rata-rata dari 141 titik ketinggian (h_i) pada jarak sampai dengan 15 km dari titik asal. Persamaan 2-2 merumuskan perhitungan EHAAT [4]:

$$h_m = \frac{\sum_{i=0}^{140} h_i}{141} \qquad 2\text{-}2$$

Tedapat dua jenis tinggi antena efektif yang digunakan: tinggi antena efektif untuk pemancar dan tinggi antena efektif untuk penerima.

2.4. ERP (*EFFECTIVE RADIATIF POWER*)

ERP merupakan daya yang dipancarkan antena setelah dikurangi *loss feeder*. ERP dinyatakan dalam satuan dBkW. Gambar 2.3 memperlihatkan perhitungan ERP.



Gambar 2.3 Perhitungan ERP

$$ERP\ (dBkW) = P_{tx} - L_{feed} + G_{antenna} \qquad 2\text{-}3$$

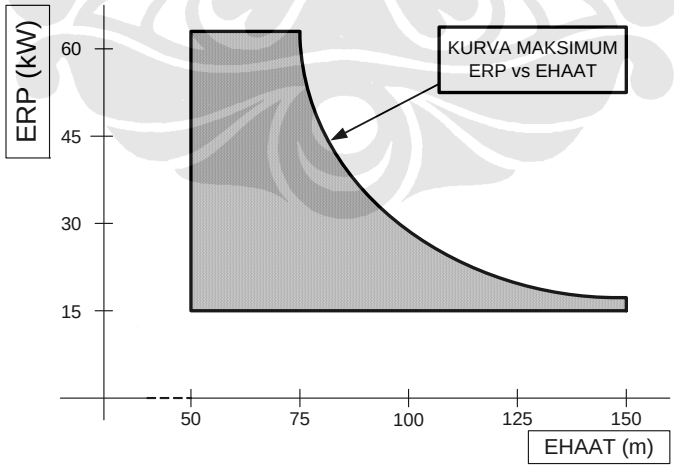
dengan

$$dBkW = 10\ Log\ (kW)$$

Loss

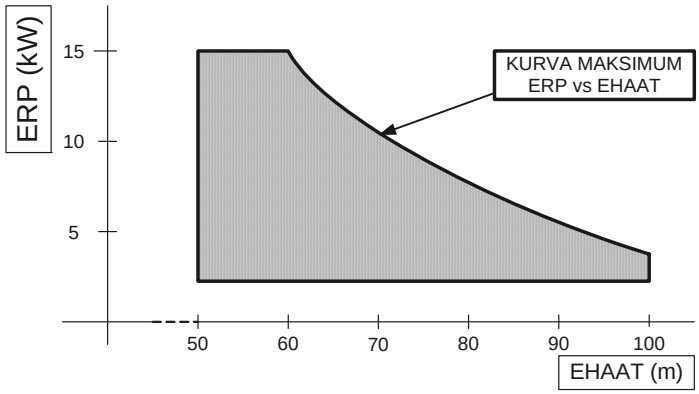
Gambar 2.4 (a) sampai dengan (d) memperlihatkan grafik hubungan antara EHAAT dengan ERP yang diambil dari KM 15:

Transmitter

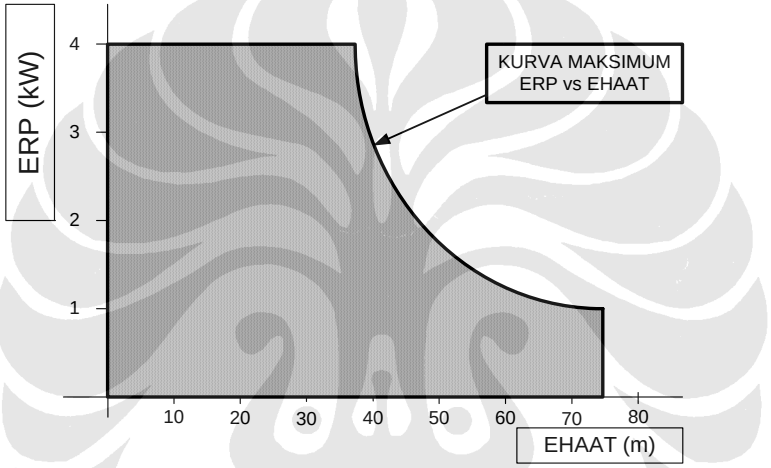


Daya transmisi TX

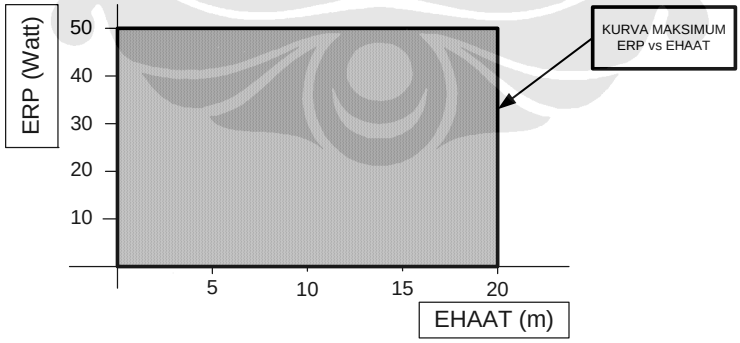
Gambar 2.4(a) Hubungan antara EHAAT dan ERP untuk Kelas A [6]



Gambar 2.4(b) Hubungan antara EHAAT dan ERP untuk Kelas B [6]



Gambar 2.4(c) Hubungan antara EHAAT dan ERP untuk Kelas C [6]



Gambar 2.4(d) Hubungan antara EHAAT dan ERP untuk Kelas D [6]

2.5. INTERFERENSI *STEADY* DAN *TROPOSPHERIC*

Seperti pada *protection ratio*, interferensi dapat dibedakan menjadi interferensi *steady* atau interferensi *tropospheric*. Maka adalah hal yang penting untuk menentukan apakah suatu interferensi digolongkan *steady* atau *tropo*. Kriteria untuk melakukan hal ini diambil dari konsep *nusance field*. *Nuisance field* adalah kuat medan pemancar yang menginterferensi pada ERP tertentu ditambah dengan *protection ratio* yang relevan[1].

Nuisance field untuk interferensi *steady* dirumuskan sebagai berikut [3]:

$$E_s = P + E(50,50) + A_s \qquad 2\text{-}4$$

Sedangkan untuk interferensi *tropospheric*:

$$E_t = P + E(50,T) + A_t \qquad 2\text{-}5$$

Dengan

P : ERP. (dB(1 kW)) dari pemancar yeng menginterferensi

A : *protection ratio* (dB)

$E(50, T)$: kuat medan (dB(μV/m)) pemancar yang menginterferensi dan ditambahkan selama T% dari total waktu. Besar T adalah 1%, yang merupakan hasil konferensi VHF/FM di Geneva tahun 1984.

Suatu interferensi disebut steady apabila hasil *nuisance field steady* lebih besar dari *nuisance field tropo*, dan sebaliknya. Penentuan jenis gangguan ini adalah penting sehingga dapat ditentukan nilai *protection ratio* mana yang harus digunakan.

2.6. *SERVICE AREA* DAN *COVERAGE AREA*

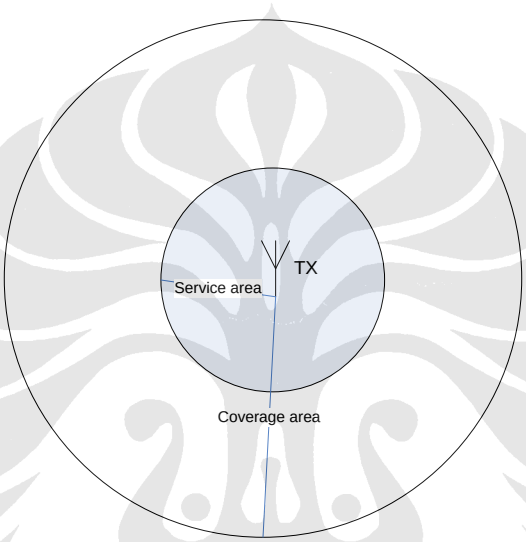
Service Area adalah wilayah cakupan layanan oleh sebuah *transmitter* tertentu dimana didalam wilayah tersebut sinyal yang dikirim dijamin dapat diterima dengan baik. Nilai *minimum field strength* pada ujung batas *service area* harus melebihi *nuisance field.*

Sedangkan *coverage area* adalah wilayah cakupan yang dapat dijangkau oleh sebuah *transmitter* dimana sinyal masih dapat diterima dengan baik, tetapi tanpa memperhatikan

[1] Diterjemahkan dari Annex 1, rekomendasi ITU-R BS.412-9

pengaruh interferensi dari pemancar lainnya. *Coverage area* adalah wilayah dimana masih diperbolehkan adanya n*uisance field* akibat pemancar lain.

Semakin besar jarak antara pemancar dan penerima, maka semakin berkurang pula nilai kuat medan pemancar tersebut. Jika suatu pemancar tidak mengalami gangguan dari pemancar lain, maka *service area* akan sama dengan *coverage area*. Apabila terdapat pemancar lain dalam wilayah yang bertetanggaan, dengan selisih frekuensi tertentu, maka akan timbul *nuisance field* di daerah *coverage area*. Gambar 2.5 memperlihatkan *service area* dan *coverage area*.



Gambar 2.5 *Service Area* dan *Coverage Area*

BAB III
BAHASA PEMROGRAMAN JAVA

3.1 Introduksi Bahasa Pemrograman Java

JAVA merupakan bahasa pemrograman berbasis *object oriented* yang diciptakan setelah C++ dan didesain sedemikian sehingga dapat beroperasi pada berbagai *platform* dan sistem operasi. Program JAVA dapat dibuat dalam bentuk *applet* yang dijalankan dengan *appletviewer* di atas *web browser*, atau dalam bentuk aplikasi mandiri yang dijalankan dengan JAVA *Interpreter*.

Sebuah aplikasi Java adalah sebuah program komputer yang menjalankan fungsinya ketika digunakan Java *command* untuk menjalankan *Java Virtual Machine* (JVM)[1].

Program-program pada Java terdiri dari bagian-bagian yang disebut kelas-kelas (*classes*). Kelas-kelas ini terdiri dari bagian-bagian yang disebut metode yang menjalankan tugas-tugas dan mengembalikan informasi ketika mereka selesai mengerjakan tugasnya.

Java sendiri telah menyediakan kelas-kelas yang mempermudah *programmer* dalam sebuah perpustakaan atau *Java class libraries*, yang biasa dikenal dengan Java APIs (*Application Programming Interfaces*)

Java memiliki beberapa keunggulan, diantaranya:

1. ***Platform independence***

   Program Java, baik *source program* maupun hasil kompilasinya tidak bergantung pada sistem operasi dan *platform* yang digunakan. Misalnya jika dibuat *source code* Java pada sistem operasi *Windows*, maka *source code* tersebut dengan mudah juga dapat dijalankan pada sistem operasi UNIX tanpa perlu mengedit kodenya sedikitpun.

   Bukan hanya *source code* yang dapat dipakai pada semua sistem operasi, hasil kompilasi Java yang merupakan *bytecode* juga dapat beroperasi pada semua sistem operasi. Biasanya, jika program ditulis pada bahasa lain (misalkan C++), maka *compiler* akan menerjemahkan *source code* ke dalam bahasa mesin yang spesifik. Hal ini menyebabkan

[1] Definisi ini diambil dari Java How to Program, 6th edition

source code perlu dikompilasi lagi, dan bahkan di-*edit* lagi untuk bisa dijalankan pada sistem lain.

Pada lingkungan Java terdapat dua bagian: *Java Compiler* dan *Java Intepreter*. *Java Compiler* mengkompilasi *source code* ke dalam *bytecode* dan memiliki ekstensi *.class*. Hasil kompilasi ini dijalankan dengan menggunakan *Java Intepreter*. *Java Interpreter* dapat dijalankan melalui *command prompt*, atau dengan *appletviewer* atau *web browser* untuk program *applet*. Hal ini menyebabkan Java dapat dijalankan di semua sistem operasi yang memiliki *Java Interpreter*.

Kelemahan menggunakan dua *layer* ini adalah kecepatan waktu yang lebih lambat.

2. Skalabilitas

Sifat *object oriented* dari Java sangat memudahkan *programmer* untuk membangun sebuah program. Ia tidak harus memulai menulis *source code* dari nol karena ia bisa menggunakan kelas-kelas yang sudah disediakan, baik oleh Java API, atau oleh *programmer* lain. Yang harus ia ketahui untuk dapat memanggil dan menggunakan sebuah kelas adalah metode dan parameter dari kelas tersebut.

Hal ini juga memudahkan *programmer* untuk mengembangkan programnya karena ia tidak harus mengubah seluruh *source code*-nya apabila ia hendak meng-*upgrade* atau mengubah program tersebut.

3.2 *OBJECT ORIENTED PROGRAMMING*

Konsep dasar dari objek dapat ditemukan dalam kehidupan sehari-hari. Manusia, binatang, gedung sekolah, telepon, kalkulator, dan lain-lain merupakan objek yang sering dijumpai sehari-hari. Demikian juga dengan program komputer. Mereka bisa dianggap juga

sebagai objek. Semua objek memiliki *atttributes* dan *behaviors*. Atribut pada objek sehari-hari misalnya warna, ukuran, bentuk, dll. Sedangkan yang menjadi *behaviors* pada objek sehari-hari, misalnya manusia, adalah berjalan, duduk, dan berdiri, dll. Semua objek, termasuk program komputer dikenali dari atribut dan *behavior* yang mereka punya.

Objek-objek yang memiliki karakteristik yang sama dapat digolongkan ke dalam kelas (*class*) yang sama. Misalnya kelas kendaraan memiliki objek-objek dengan karakteristik yang mirip seperti mobil, truk, sepeda, pesawat, kapal laut dll. Sebuah kelas juga bisa memiliki subkelas, yang merupakan turunan (*inheritance*) dari kelas yang diatasnya (*superclass*). Misalnya kelas kendaraan dapat memiliki subkelas kendaraan darat, kendaraan laut, dan kendaraan udara. Sub kelas memiliki karakteristik yang lebih unik dari *superclass*-nya.

Java merupakan bahasa berbasis *object oriented*. Hal ini berarti unit-unit program dari program Java adalah kelas-kelas yang didalamnya objek-objek diciptakan. Kelas-kelas Java terdiri dari *methods* (operasi atau fungsi dari kelas tersebut), dan *fields* (atribut-atribut dari kelas). Metode-metode tersebut akan memanipulasi atribut-atribut dan memberikan pelayanan pada kelas lain yang memakainya (*clients*). Tugas *programmer* adalah membuat kelas-kelas yang saling berinteraksi satu dengan yang lainnya, dengan sebuah kelas utama (*main class)* dimana kelas-kelas diikat menjadi sebuah kesatuan sehinga menjadi sebuah program yang utuh.

Untuk berinteraksi dengan kelas-kelas yang lain, kelas utama tidak perlu untuk mengetahui detail bagaimana mereka mengerjakan tugasnya. Kelas utama tersebut hanya perlu mengetahui operasi apa yang mereka kerjakan, dan apa yang menjadi atributnya. Hal inipun terjadi dalam kehidupan sehari-hari. Misalkan dalam mengendarai mobil, pengendara tidak perlu tahu detail bagaimana alat-alat dalam mobil bekerja (pedal gas, pedal rem, stir, dll). Ketika pedal gas ditekan, maka pengendara memberikan *method call* dan menyuruh sebuah metode dari objek tersebut untuk melaksanakan tugasnya. Mobil juga memiliki atribut-atribut seperti warna mobil, jumlah pintu, banyak bensin, kecepatan, dll. Dalam pemrograman Java, atribut-atribut diwakili oleh *instance variables.*

3.2.1 *Classes, Objects,* and *Inheritance*

Paradigma dari *Object Oriented* ditemukan dalam relasi antara kelas, objek, dan *inheritance.*

Classes

Kelas mendefinisikan struktur dari objek yang dimilikinya. Sebuah kelas terdiri dari variabel-variabel (data) dan metode-metode (*actions*) yang menentukan *behavior* dari sebuah objek[2]. Metode-metode dalam sebuah kelas dapat dideklarasikan `public` ataupun `private`. Metode publik dapat diakses oleh metode lain diluar kelasnya, sedangkan metode privat hanya bisa diakses di dalam kelas tersebut. Sebuah method juga dapat didesain untuk mengambalikan nilai tertentu atau tidak mengambalikan nilai apapun. Metode yang menjalankan prosesnya dan mengembalikan nilai tertentu dideklarasikan dengan tipe nilai yang akan dikembalikan. Metode yang tidak mengambalikan nilai apapun dideklarasikan dengan `void`. Format dasar dari sebuah metode adalah:

```
   tipe-nilai-yang-dikembalikan nama( argumen1, argumen2, .. )
{
     ...kode program...
}
```

Gambar 3.1 memperlihatkan contoh kode Java kelas `Titik` yang digunakan untuk memanipulasi objek-objek dari `Titik`.

[2] Diambil dari Object Oriented Data Structure Using Java, Jones and Bartlett publisher

```
 1  public class Titik
 2  {
 3
 4  protected int koorX;
 5  protected int kooY;
 6
 7  public Titik( int X, int Y )
 8  // Inisialisasi Titik dengan parameter-parameter
 9  // koorX dan koorY
10  {
11
12  koorX = X;
13  koorY = Y;
14
15  }
16
17  public int koordinatX()
18  {
19  return koorX;
20  }
21
22  public int koordinatY()
23  {
24  return koorY;
25  }
26  }
```

Gambar 3.1 Contoh Kelas `Titik`

Kelas `Titik` diatas memiliki *instance variable:* `koorX`, dan `koorY`. Nilai dari *instance variable* bervariasi untuk setiap objekdari kelas, nilai ini mewakili atribut-atribut dari kelas `Titik`.

Ada tiga metode dalam kelas `Titik` diatas: `Titik`, `koordinatX` dan `koordinatY`. Metode `Titik` memiliki nama yang sama dengan nama kelas, dan merupakan *constructor* dari kelas `Titik`. Tugas *constructor* dari sebuah kelas adalah untuk meng-instansiasi objek dari kelas. Tiga metode yang lain (`koordinat` dan `koordinatY`) merupakan *observer* karena mereka melakukan fungsi observasi dan mengembalikan nilai dari *instance variable* `koorX` dan `koorY`.

Object

Objek diciptakan dari kelas-kelas ketika program dijalankan. Sistem *Object Oriented* harus dilihat sebagai kumpulan objek-objek yang saling bekerja satu dengan yang lainnya untuk menyelesaikan suatu tugas.

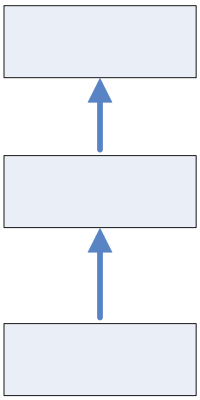
Untuk membuat objek dari suatu kelas, digunakan operator `new`, kemudian dilanjutkan dengan *constructor* dari kelas. Dibawah ini adalah sintaks pembuatan objek `Titik_1`, `Titik_2`, dan `Titik_3` dari kelas `Titik.` `Titik_1` merupakan sebuah titik dengan koordinat kartesius (12,15), `Titik_2` merupakan sebuah titik dengan koordinat kartesius (1,2) `Titik_3` merupakan sebuah titik dengan koordinat kartesius (2,6),).

```
Titik Titik_1 = new Titik ( 12, 15 );
Titik Titik_2 = new Titik ( 1, 2 );
Titik Titik_3 = new Titik ( 2, 6 );
```

Inheritance

Salah satu keunggulan *Object Oriented* adalah dapat digunakannya kembali kelas-kelas yang ada. Hal ini dikenal dengan *inheritance*. *Inheritance* memampukan *programmer* untuk membuat kelas baru yang lebih spesifik dan spesial dari kelas yang telah ada. Kelas yang baru ini disebut *subclass*, sedangkan kelas yang sebelumnya disebut *superclass*. *Subclass* menurunkan sifat-sifat dari *superclass*, dan menambahkan dengan sifat-sifat yang baru yang lebih spesifik.

Sekarang akan dibuat kelas baru (kelas `Lingkaran` dan kelas `Tabung`) yang mengambil fitur-fitur dari kelas-kelas yang telah dbuat sebelumnya. Kelas Lingkaran merupakan subkelas dari kelas `Titik`, dan kelas `Tabung` merupakan subkelas dari kelas `Lingkaran`. Gambar 3.2 memperlihatkan diagram pohon dari ketiga kelas tersebut. Gambar 3.3 memperlihatkan kode kelas `Lingkaran`, dan Gambar 3.4 memperlihatkan kode dari kelas `Tabung`.



Gambar 3.2 Diagram kelas `Titik`, `Lingkaran`, dan `Tabung`

```
 1  public class Lingkaran extends Titik
 2  {
 3
 4  protected int radius;
 5
 6  public Lingkaran ( int X, int Y, int R )
 7  {
 8  super ( X, Y );
 9  radius = R;
10  }
11
12  public int jari-jari()
13  {
14  return radius;
15  }
16
17  }//akhir dari kelas Lingkaran
```

Gambar 3.3 Kode untuk Kelas `Lingkaran`

```
 1  public class Tabung extends Lingkaran
 2  {
 3
 4  protected double tinggi;
 5  private double volum;
 6
 7  public Lingkaran ( int X, int Y, int R, double T)
 8  {
 9  super ( X, Y , R);
10  tinggi = T;
11  }
12
13  public double tinggiTabung()
14  {
15  return tinggi;
16  }
17
18  public double volumTabung()
19  {
20  volum = 3.14*R*R*T;
21  Return volum;
22
23  }
24  }//akhir dari kelas Lingkaran
```

Gambar 3.4 Kode untuk Kelas `Tabung`

Semua kelas yang ada pada Java (kecuali kelas `Object`) merupakan perpanjangan (*extends*) kelas yang sudah ada. Kelas `Titik` merupakan turunan dari atau memperpanjang kelas `Object` (dari *package* `Java.lang`). Secara langsung atau tidak langsung, semua kelas merupakan turunan menurunkan metode-metode yang ada pada kelas `Object`. Apabila sebuah kelas tidak secara explisit memperpanjang kelas lainnya, maka secara implisit kelas tersebut memperpanjang kelas `Object`. *Programmer* biasanya tidak mencantumkan "`extends Object`" pada kodenya.

Kelas `Lingkaran` merupakan subkelas dari kelas `Titik`. *Constructor* dari subkelas (`Lingkaran`) harus memanggil *constructor* dari *superclass*-nya (`Titik`) sehingga *instance variable* yang diturunkan dari *superclass* (`X` dan `Y`) dapat diinisialisasikan dengan tepat. Hal ini dinyatakan dengan "`super`". Perhatikan bahwa pada kelas `Lingkaran` terdapat tambahan variabel yang tidak terdapat pada kelas `Titik`, yaitu radius. Hal ini dikarenakan lingkaran tidak hanya memiliki titik pusat dengan koordinat x dan y, tetapi juga panjang jari-jari lingkaran.

Kelas `Tabung` merupakan subkelas dari `Lingkaran`, dimana metode dan *instance variable* dari kelas `Lingkaran` diturunkan pada kelas `Tabung`, dengan ditambah variabel baru, yaitu tinggi tabung.

Ketika menggunakan *inheritance* dalam melakukan pemrograman, subkelas bisa mengakses baik metode dan *isntance variable* dari *superclass*-nya, tetapi perhatikan bahwa *superclass* tidak dapat mengakses metode dan *instance variable* dari subkelasnya. *Compiler* akan menyatakan *error* ketika *superclass* mencoba mengakses subkelasnya.

3.2.2 *Polymorphism*

Salah satu bentuk *object oriented* adalah *polymorphism*. *Polymorphism* memampukan *programmer* untuk membuat program secara umum dari pada secara khusus. Pada intinya, *polymorphism* memampukan *programmer* untuk membuat program yang memproses objek-objek yang dengan *superclass* yang sama.

Misal kita akan merancang sebuah program untuk menghitung luas volum bangun-bangun ruang, meliputi tabung dan kerucut. Kedua-duanya memiliki *instance variable* yang sama, yaitu diameter dan tinggi. Kita membuat kelas `BangunRuang` yang memiliki metode `Volume`, dan menjadi *superclass* untuk kelas `tabung` dan `kerucut`. Semua bangun subkelas mengimplementasi metode `Volume` dari kelas `BangunRuang`. Ketika simulasi dijalankan, program akan mengirimkan pesan umum untuk menghitung luas volum pada kelas `tabung` dan `kerucut`, tetapi baik `tabung` dan `kerucut` akan berespon secara unik (rumus luas volum yang berbeda). Pesan yang sama, dalam hal ini menghitung luas volum, tetapi ditanggapi secara bervariasi (*many forms*). Itulah mengapa cara ini disebut *polymorphism*.

Metode dan Kelas Abstrak

Salah satu cara untuk melakukan *polymorphism* adalah dengan membuat metode dan kelas abstrak. Kelas dan metode ini tidak dapat digunakan untuk menciptakan objek, sehingga semua subkelas harus “mengisi kekurangan” dari kelas dan metode abstrak. Tujuan dari kelas dan metode abstrak adalah untuk menyediakan kelas dan metode yang tepat sedemikian sehingga semua kelas yang lain dapat menjadi subkelasnya.

Gambar 3.5 sampai Gambar 3.7 memperlihatkan kode untuk kelas `BangunRuang`, `Tabung`, dan `Kerucut`. Kelas `BangunRuang` merupakan kelas abstrak, dan kelas `Tabung` dan `Kerucut` merupakan subkelas dari `BangunRuang`.

```
 1  public abstract class BangunRuang
 2  {
 3  private int diameter;
 4  private int tinggi;
 5  public BangunRuang( int d, int t )
 6  {
 7  diameter = d;
 8  tinggi = t;
 9  }
10  public void setDiameter ( int d )
11  {
12  diameter = d;
13  }
14  public int getDiameter()
15  {
16  return diameter;
17  }
18  public void setTinggi ( int t )
19  {
20  tinggi = t;
21  }
22  public int getTinggi()
23  {
24  return tinggi;
25  }
26  public abstract double Volume(); //metode abstrak
27  }//akhir dari kelas abstrak BangunRuang
```

Gambar 3.5 Kode Kelas `BangunRuang`

Baris pertama merupakan deklarasi kelas dimana kelas dengan nama `BangunRuang` merupakan kelas abstrak. Baris 5 sampai 9 merupakan konstruktor dari kelas `BangunRuang` yang memiliki dua argumen (diameter dan tinggi). Baris ke-26 menyatakan metode abstrak yang nantinya akan didefinisikan oleh setiap subkelas.

```
 1   public class Tabung extends BangunRuang
 2   {
 3   private double volume;
 4   private int r;
 5   public Tabung ( int d, int t )
 6   {
 7   super ( d, t );
 8   }
 9   public double getVolume()
10  {
11  r = 0.5* d;
12  volume = 3.14*r*r*t;
13  return volume;
14  }
15  public double Volume()
16  {
17  return getVolume();
18  }
19  }//akhir dari kelas Tabung
```

Gambar 3.6 Kode Kelas `Tabung`

```
 1   public class Kerucut extends BangunRuang
 2   {
 3   private double volume;
 4   private int r;
 5   public Kerucut ( int d, int t )
 6   {
 7   super ( d, t );
 8   }
 9   public double getVolume()
10  {
11  r = 0.5* d;
12  volume = 3.14*r*r*t;
13  return volume;
14  }
15  public double Volume()
16  {
17  return getVolume();
18  }
19  }//akhir dari kelas Tabung
```

Gambar 3.7 Kode Kelas `Kerucut`

Kelas `Tabung` dan `Kerucut` mendefinisikan metode abstrak `Volume()` dari kelas `BangunRuang`, sehingga kelas `Tabung` dan `Kerucut` memberikan respon yang berbeda (Gambar 3.6 dan 3.7 baris ke-12) terhadap metode `Volume()`.

3.3 *CONTROL STATEMENT*

Ketika merancang sebuah program, adalah hal yang penting untuk mengerti bagaimana suatu permasalahan dianalisa dan diselesaikan. Pengertian yang baik terhadap *control statements* membantu *programmer* untuk memecahkan masalah dengan memanipulasi objek, yaitu dengan mengontrol urutan aksi-aksi yang ada pada program. Subbab 3.4 ini akan membahas beberapa dari *control statement* yang sering digunakan dalam melakukan pemrograman.

Algoritma

Algoritma adalah logika berpikir yang digunakan untuk menyelesaikan suatu masalah dengan memperhatikan dua hal: aksi yang dilakukan dan urutan bagaimana aksi-aksi tersebut dikerjakan. Misalkan apabila kita hendak membuat program yang mampu menukar isi dua buah tempat penyimpanan (misal A dan B), maka kita dapat membuat algoritma sebagai berikut: pindahkan isi A ke C (tempat sementara), pindahkan isi B ke A, dan akhirnya pindahkan isi C ke B. Sebuah algoritma tidak terpaku pada satu bahasa pemrograman tertentu, dan biasanya ditulis dengan *pseudocode*.

Pseudocode

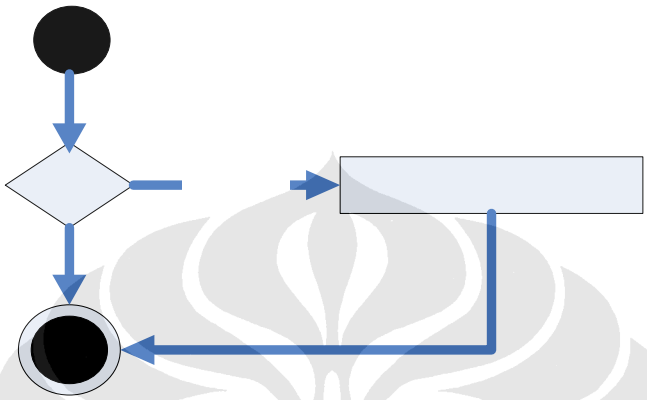
Pseudocode merupakan bahasa informal yang digunakan untuk mengungkapkan sebuah algoritma. *Pseudocode* tidak terpaku pada satu sintaks bahasa pemrograman tertentu sehingga dapat dibuat dengan bahasa yang mudah dan sederhana. Setelah *pseudocode* dibuat, maka *programmer* akan mengkodekannya kedalam bahasa pemrograman tertentu.

if

Pernyataan `if` akan menyebabkan suatu atau beberapa aksi akan dilakukan dengan kondisi tertentu yang sudah dipenuhi. Misal jika kita menghendaki siswa yang nilainya dibawah 4.0 tidak lulus, maka didalam sintaks Java dapat ditulis

```
if ( nilai < 4.0 )
    System.Out.println( ”tidak lulus” );
```

Sintaks diatas berarti bahwa apabila nilai seorang siswa lebih kecil dari 4.0, maka program akan menampilkan tulisan ”tidak lulus”. Apabila nilai siswa diatas atau sama dengan 4.0, maka program akan melanjutkan ke aksi berikutnya tanpa menampilkan tuilisan apapun. Diagram yang menyatakan logika pernyataan `if` diatas diperlihatkan pada Gambar 3.8.



Gambar 3.8 Diagram Pernyataan `if`

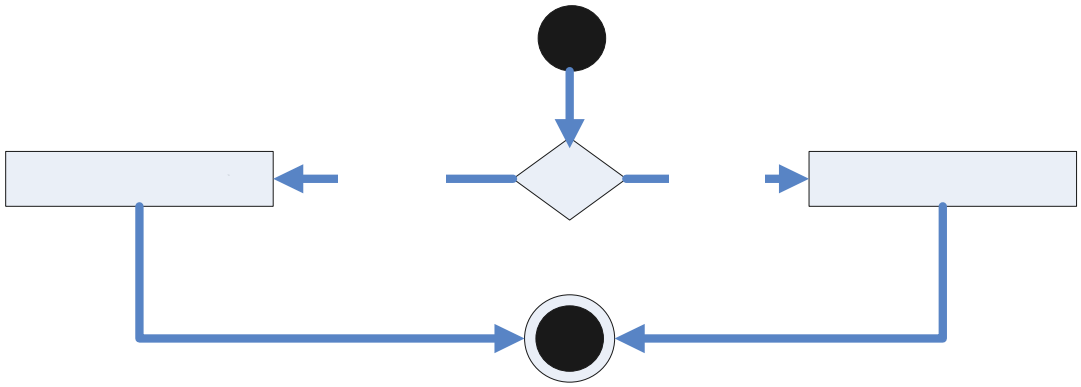
`if…else`

Pernyataan `if…else` menyebabkan beberapa aksi dikerjakan untuk beberapa kondisi yang berbeda. Misal jika siswa mendapat nilai kurang dari 4.0 dinyatakan tidak lulus, sedangkan mereka yang mendapat nilai diatas atau sama dengan 4.0 dinyatakan lulus dapat ditulis sebagai berikut.

Nilai < 4.0

```
if ( nilai < 4.0 )
    System.Out.println( ”tidak lulus” );
else
    System.Out.println( ”lulus” );
```

Diagram untuk logika `if...else` diperlihatkan pada Gambar 3.9.



Gambar 3.9 Diagram Pernyataan `if…else`

`while`

Cetak “ lulus”

Nilai >= 4.0

Pernyataan `while` merupakan pernyataan repetisi karena menyebabkan sebuah atau beberapa aksi terus menerus dikerjakan selama masih memenuhi kondisi tertentu. Misalkan sebuah program untuk menghitung bilangan m pangkat n dapat ditulis sebagai berikut.

```
int n, m, hasil;
while ( n =! 1 ){
    hasil = m*m;
    n = n -1; }
```

Sintaks diatas menyatakan bahwa program akan terus mengalikan bilangan m dengan dirinya selama bilangan n bernilai tidak sama dengan 1, dan program baru akan melanjutkan aksi berikutnya.

`for`

Pernyataan `for` juga merupakan pernyataan repetisi, sama seperti `while`. Misalkan sebuah sintaks untuk menampilkan isi larik (*array*) dapat ditulis sebagai berikut.

```
String output = ””;
int larik[] = new int[ 3 ];
larik = { SD, SMP, SMU };
for ( int count = 0; count < larik.length; count ++ )
    output = output + larik[count];
```

```
System.Out.println( output );
```

Kode diatas adalah syntax untuk menampilkan isi `larik[]` (SD, SMP, dan SMA) pada layar monitor. Apabila nilai `count` belum sama dengan jumlah komponen `larik[]` (`larik.length`), maka satu demi satu isi tiap komponen larik disimpan dalam variabel `output`. Setelah semua isi komponen `larik[]` disimpan, maka program akan menampilkan `output` pada layar monitor.

BAB IV
PERANCANGAN *SOFTWARE*

4.1 *CLASSES* DAN *METHODS*

4.1.1 *CLASSES*

Seperti yang telah dejelaskan pada Bab 3, program berbasis Java terdiri dari kelas-kelas yang nantinya dipanggil oleh kelas utama (*main class*) ketika program dijalankan. Sebuah kelas dapat berdiri sendiri, merupakan turunan dari kelas sebelumnya, atau memanggil kelas lainnya. Tabel 4.1 memperlihatkan daftar kelas-kelas yang ditulis pada program optimalisasi kanal radio siaran FM ini beserta masing-masing fungsi yang dikerjakannya.

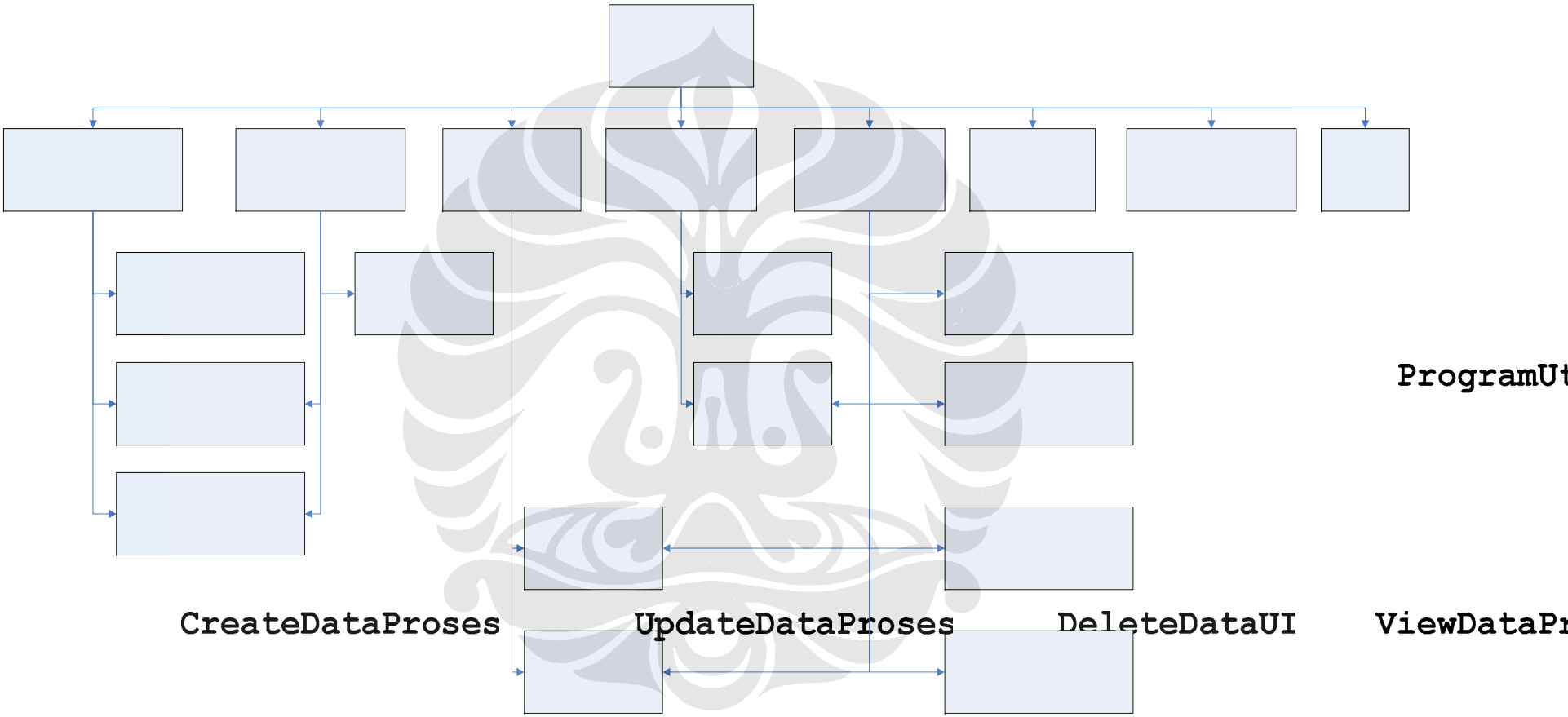
Tabel 4.1 Daftar Kelas-kelas

Nama Kelas	Fungsi
`CreateDataProses`	Menjalankan proses membuat data baru
`CreateDataUI`	*User Interface* untuk membuat data baru.
`DeleteDataUI`	*User Interface* untuk menghapus data.
`HelpGeneralInterface`	*User Interface* untuk menampilkan pesan *help* secara umum.
`HelpDataInterface`	*User Interface* untuk menampilkan pesan *help menu data* .
`HelpOptimalisasiInterface`	*User Interface* untuk menampilkan pesan *help* menu optimalisasi FM.
`KanalEditor`	*Input* nomor-nomor kanal FM
`LongLatInterface`	*User Interface* untuk memasukkan nilai lintang dan bujur pemancar
`OpInterface`	*User Interface* untuk melakukan optimalisasi distribusi nomor kanal
`OptimalisasiKanal`	Menjalankan fungsi optimalisasi FM

`PilihDataKonsider`	Memilih pemancar-pemancar yang harus dikonsider, yaitu mereka yang berada dalam radius 200km dengan pemancar *wanted*.
`ProgramUtama`	*User Interface* untuk tampilan utama program
`SusunPemancar`	Menyusun pemancar-pemancar dalam satu propinsi
`TestDelete`	Menghapus data
`TestSearch`	Mencari data berdasarkan nama pemancar
`TestUpdate`	Mengubah data lama dengan data baru
`TestWrite`	Menulis data baru
`UpdateDataProses`	Menjalankan proses *update* data
`UpdateDataUI`	*User Interface* untuk mengubah data lama dengan data baru
`ViewDataProses`	Menjalankan proses untuk melihat daftar pemancar dalam satu propinsi yang sudah disimpan berikut nilai lintang, bujur, dan nomor-nomor kanal.
`ViewDataUI`	*User Interface* untuk melihat daftar pemancar dalam satu propinsi yang sudah disimpan berikut nilai lintang, bujur, dan nomor-nomor kanal

Disamping kelas-kelas yang terdaftar dalam Tabel 4.1, terdapat juga kelas-kelas yang berfungsi untuk menampilkan jam *digital* secara *real time* yang didapat dari internet. Kelas-kelas ini merupakan *free class* sehingga dapat dipergunakan oleh siapa saja dengan mencangkokkannya ke dalam program. Kelas-kelas tersebut adalah `ImageColon`, `ImageDigits`, `ImageLeftBorder`, `ImageSplash`, `ImageSpace`, `LcdClock`, `LcdClockPanel`, `LcdTable`.

Gambar 4.1 memperlihatkan hubungan antar kelas yang terdapat pada program optimalisasi kanal radio siaran FM ini.



Gambar 4.1 Diagram Kelas-kelas

Kelas `ProgramUtama` memanggil kelas-kelas `CreateDataProses`, `UpdateDataProses`, `DeleteDataUI`, `HelpGeneralInterface`, `HelpDataInterface`, `HelpOptimalisasiInterface`, dan `ViewDataProses`. Kelas `CreateDataProses` menjalankan proses untuk membuat data baru, termasuk didalamnya menampilkan *user interface* dengan memanggil kelas `CreateDataUI` dan `LongLatInterface`. Kelas `UpdateDataProses` menjalankan proses untuk meng-*update* data lama dengan data baru, termasuk didalamnya menampilkan *user interface* dengan memanggil kelas `UpdateDataUI` dan `LongLatInterface`

4.1.2 *METHODS*

Setiap kelas memiliki metode-metode yang berfungsi untuk menjalankan fungsi kelas tersebut. Tabel 4.2 memperlihatkan daftar metode-metode yang terdapat pada masing-masing kelas beserta deklarasi dan argumen-argumennya.

Tabel 4.2 Daftar Metode-metode pada tiap-tiap kelas

Nama Kelas	**Nama Metode**	***Return-type***	**Argumen**	**Fungsi**
`CreateDataUI`	`create`	`void`	-	Membuat data dengan memanggil metode addData() pada kelas TestWrite.
	`addComponent`	`void`	`Component component, int row, int column, int width, int height`	Menambah komponen GUI (*Graphic User Interface*) yang dinyatakan dalam *constructor*
	`getcreateButton, getHelpButton, getClearButton,getInputLongLatButton,getInputKanalButton.`	`JButton`	-	Mengambalikan objek masing-masing Jbutton
	`getPropinsiField, getNamaField,getLongField,getLatField,getKanalArea.`	`String`	-	Mengambalikan apa yang tertulis pada objek JTextField dan JTextArea
	`setPropinsiField, setNamaField, setLongField, setLatField, setKanalArea`	`void`	`String input`	Mengisi objek JTextField dan JTextArea dengan teks.

	clearFields	void	-	Menghapus apa yang tertulis pada semua objek JTextField dan JTextArea
DeleteDataUI	pencarian	void	-	Mencari data dan menampilkannya pada *interface*
	penghapusan	void	-	Menghapus data yang sudah dicari
	addComponent	void	Component component, int row, int column, int width, int height	Menambah komponen GUI (Graphic *User* Interface) yang dinyatakan dalam *constructor*
	getDeleteButton, getHelpButton, getClearButton,.	JButton	-	Mengambalikan objek masing-masing JButton
	getPropinsiField, getNamaField,getLongField,getLatField,getKanalArea.	String	-	Mengambalikan apa yang tertulis pada objek JTextField dan JTextArea
	setPropinsiField, setNamaField, setLongField, setLatField, setKanalArea	void	String input	Mengisi objek JTextField dan JTextArea dengan teks.
	clearFields	void	-	Menghapus apa yang tertulis pada semua objek JTextField dan JTextArea
KanalEditor	inputNomorKanal	void	-	Memasukkan *input* nomor kanal
	getNoChannel	Int[]	-	Mengambil *array* nomor kanal dari

				metode inputNomorKanal()
LongLatInterface	addComponent	void	Component component, int row, int column, int width, int height	Menambah komponen GUI (Graphic *User* Interface) yang dinyatakan dalam *constructor*
	getOkButton	JButton	-	Mengembalikan objek JButton
	getFieldLongDeg, getFieldLongMin, getFieldLongSec, getFieldLatDeg, getFieldLatMin, getFieldLatSec	int	-	Mengembalikan *input* lintang dan bujur
OpInterface	inputSpasi	Void	-	Memasukkan *input* spasi antar pemancar
	op2	void	Int jmlPemancar, int nomorKanalTerkecil	Melakukan *looping* untuk proses sebanya jumlah pemancar yang dikonsider untuk proses optimalisasi kanal
	addComponent	void	Component component, int row, int column, int width, int height	Menambah komponen GUI (*Graphic User Interface*) yang dinyatakan dalam *constructor*
	clearFields	void	-	Menghapus apa yang tertulis pada semua objek JTextField dan JTextArea
OptimalisasiKanal	hitungJumlahKanal	Int	String baris	Menghitung berapa jumlah kanal yang terdapat pada satu pemancar
	baca*File*	String	String nama*File*	Membaca seluruh isi *file* sampai eof (end of *file*) dan menyimpannya dalam bentuk

				string
	readFile	String	String namaFile	Membaca seluruh isi *file* sampai "eof' kemudian menyimpannya ke dalam string
	optimalisasiFM	String	String sumber, String fill, int spasi, int jmlKanal	Melakukan proses pencarian nomor kanal untuk pemancar tujuan
PilihDataKonsider	op1	void	String nama*File*, String dataSasaran	Memilih pemancar yang harus dikonsider dan menyimpannya dalam *file* daftarKonsider.har
SusunPemancar	hitungNamaPemancar	String	String nama*File*	Menghitung berapa banyak jumlah pemancar yang ada dalam sebuah *file*
	susunNamaPemancar	String	String nama*File*	Menyusun nama-nama pemancar dalam satu *file* ke dalam string
TestDelete	deleteAll	Void	String nama*File*	Menghapus semua data dalam satu *file*
	deleteData	Void	String nama*File*, String input	Menghapus data tertentu (input) dalam sebuah *file* (nama*File*)
TestSearch	cariBarisData	String	String nama*File*, String input	Mencari baris didalam sebuah *file* (nama*File*) dimana data yang hendak dicari (input) disimpan.
	tampilPerBaris	String	String input	Menampilkan data yang telah ditemukan
	cariData	String	String nama*File*, String input	Mencari data tertentu dalam sebuah *file*
TestUpdate	deleteData	Void	String nama*File*, String input	Menghapus data tertentu (*input*) dalam sebuah *file* (nama*File*)

	write*File*	void	String nama*File*, String data	Menyimpan seluruh data pada sebuah *file*
	read*File*	String	String nama*File*	Membaca seluruh data dan memindahkannya ke dalam String
	addData	void	String nama*File*, String dataBaru	Menambahkan data baru pada sebuah *file*
	updateData	void	String nama*File*, String dataLama, String dataBaru	Mengubah data lama dengan data baru
TestWrite	bacaBaris	String	String nama*File*	Membaca baris pertama dari *file*
	write*File*	void	String nama*File*, String data	Menyimpan seluruh data pada sebuah *file*
	read*File*	String	String nama*File*	Membaca seluruh data dan memindahkannya ke dalam String
	addData	void	String nama*File*, String dataBaru	Menambahkan data baru pada sebuah *file*
UpdateDataUI	updating	void	-	Melakukan proses *update* data
	pencarian	Void	-	Mencari data yang akan di-*update*
	addComponent	void	Component component, int row, int column, int width, int height	Menambah komponen GUI (*Graphic User Interface*) yang dinyatakan dalam *constructor*
	getUpdateButton, getHelpButton, getClearButton,ge	JButton	-	Mengambalikan objek masing-masing JButton

	tInputLongLatButton,getInputKanalButton.			
	getPropinsiField, getNamaField,getLongField,getLatField,getKanalArea.	String	-	Mengambalikan apa yang tertulis pada objek JTextField dan JTextArea
	setPropinsiField, setNamaField, setLongField, setLatField, setKanalArea	void	String input	Mengisi objek JTextField dan JTextArea dengan teks.
	clearFields	void	-	Menghapus apa yang tertulis pada semua objek JTextField dan JTextArea

4.2 MANIPULASI DATA

Untuk dapat melakukan proses optimalisasi distribusi kanal, yaitu pencarian nomor-nomor kanal yang masih dapat dialokasikan bagi suatu daerah, maka diperlukan sebuah sarana penampungan data yang darinya segala informasi yang dibutuhkan bisa didapat. Informasi tersebut berupa; nama pemancar, nilai lintang, nilai bujur, dan nomor-nomor kanal yang sudah dialokasikan untuk pemancar tersebut. Berarti setiap pemancar akan memiliki variabel sebanyak jumlah kanal + 3. Setiap beberapa pemancar akan disimpan di dalam satu *file* yang sama, yaitu berdasarkan propinsi. Jadi semua kota dalam satu propinsi akan dikumpulkan di dalam satu *file* propinsi tersebut. Nama pemancar disesuaikan dengan nama kota atau daerah. Format data yang disimpan adalah sebagai berikut:

Nama-pemancar;lintang;bujur;nomor-kanal1;nomor-kanal2;…

Setiap variabel dipisahkan dengan tanda titik koma (;) sehingga *user* tidak boleh menggunakan tanda ini ketika memasukkan *input* karena dapat menyebabkan program mengalami kesalahan ketika mengambil variabel yang ia butuhkan. Digunakan kelas `StringTokenizer` dengan *constructor* `StringTokenizer( String str, String delim )` dari *package* `java.util.StringTokenizer` untuk memecah satu baris menjadi partisi-partisi berdasarkan *delimiter* yang diberikan. *Delimiter* untuk program ini adalah titik koma (";"). Metode `nextToken()` yang mengembalikan nilai string digunakan untuk mengambil variabel-variabel dari baris data. Berikut adalah sintaks yang digunakan dalam perancangan program ini:

```
StringTokenizer token = new StringTokenizer( String input, ";");
String namaPemancar = token.nextToken();
String lintang = token.nextToken();
String latitude = token.nextToken();
String kanal1 = token.nextToken();
While( token.hasMoreTokens() )
{
  String kanal = token.nextToken();
```

```
  String kanal1 = kanal1 + ";" + kanal;
}
```

Sintaks diatas membagi satu baris data pemancar kedalam empat bagian: nama pemancar, nilai lintang, nilai bujur, dan daftar nomor kanal yang juga dipisahkan dengan tanda titik koma. Ketika diperlukan, daftar nomor kanal ini nantinya akan dipisahkan lagi dan dimasukkan ke dalam *array* sehingga dapat diproses.

Sebagai tempat menyimpan data, digunakan objek dari kelas `File` dari *package* `java.io.File` yang akan membuat sebuah *file* dengan nama sesuai parameter nama *file* yang akan dibuat. Untuk membaca dan menulis data ke dalam *file*, digunakan objek dari kelas `FileReader` dan `FileWriter` dari package `java.io.FileReader` dan `java.io.FileWriter`. Objek dari kelas `BufferedReader` dan `BufferedWriter` digunakan untuk menyediakan buffer ketika akan menulis atau membaca data dari `FileReader` dan `FileWriter`.

4.2.1 *Write Data*

Proses penulisan data ke dalam sebuah *file* dilakukan oleh kelas `TestWrite` dan meliputi:

1. Pembacaan seluruh *file*
2. Pindahkan isi *file* ke dalam sebuah string dengan metode `readLine`() dari kelas `BufferedReader`.
3. Menambahkan data baru ke dalam string
4. Menuliskan string yang sudah ditambah data baru ke *file*. Data yang sebelumnya akan diganti dengan data yang baru, sehingga data tidak akan tersimpan dua kali. Gambar 4.2, 4.3, dan 4.4 berturut-turut memperlihatkan kode pada metode `write`*`File`*, `readFile`, dan `addData` untuk menambah data baru pada *file*,

```
public static void writeFile(String namaFile, String data)
{
     try
    {
      File out = new File(namaFile);
      FileWriter fileOut = new FileWriter(out);
      BufferedWriter fileWriteOut = new BufferedWriter(fileOut);
      fileWriteOut.write(data + "\n" + "eof" );
      fileWriteOut.close();
    }
    catch(Exception e)
   {
      JOptionPane.showMessageDialog( “error writing file” );
   }
}
```

Gambar 4.2 Kode Metode `write*File*( String namaFile, String data)`

```
public static String readFile(String namaFile)
{
String fill = "";
String data = "";
try
{
      File file=new File(namaFile);            FileReader reader=new
FileReader(file);
      BufferedReader baca=new BufferedReader(reader);
      fill=baca.readLine();
      if ( fill.equals("") )
            data = "";
      else
      {
            data = fill;
            while (!(fill.equals("eof")))                              {
                  fill = baca.readLine();

            if(!(fill.equals("eof")))
                  {
                        data = data + "\n" + fill;
                  }
            }
      }
}
catch(Exception e)
{
   JOptionPane.showMessageDialog( “error writing file” );
}
return data;
}//END OF METHOD
```

Gambar 4.3. Kode Metode `read*File*(String nama*File*)`

Pada Gambar 4.3 diatas, perintah

```
     if ( fill.equals("") )
data = "";
```

berlaku apabila data baru merupakan data pertama yang ditulis. Sehingga data lama akan berisi *null* dan isi data baru akan sama dengan data lama. Metode `readFile` mengembalikan sebuah string, yaitu isi dari *file* yang dibaca. Pada metode `writeFile` (Gambar 4.2), variabel string yang hendak ditulis ke dalam *file* ditambahkan dengan string "eof" (singkatan dari *end of file*). String ini digunakan sebagai penanda bahwa data sudah habis. Semua metode dalam memanipulasi data yang ada pada program ini akan medeteksi keberadaan "eof". Apabila "eof" dideteksi, maka suatu manipulasi data yang sedang berlangsung akan selesai.

```
public static void addData(String namaFile, String
dataBaru)
     {
          String dataTotal = "";
          String NAMAFILE = namaFile + ".har";

          String dataLama = readFile(NAMAFILE);

          if ( dataLama.equals(""))
               dataTotal = dataBaru;
          else
               dataTotal = dataLama + "\n" +dataBaru;

          writeFile(NAMAFILE, dataTotal);
     }
```

Gambar 4.4. Kode Metode `addData( String namaFile, String dataBaru)`

***Delete* Data**

Proses penghapusan data dari sebuah *file* pada program ini dilakukan oleh kelas `TestDelete`. *Combo box* pemancar pada *user interface* `deleteDataUI` berisi semua data yang terdapat pada suatu propinsi. Algoritma penghapusan data adalah dengan cara menulis ulang seluruh data pada suatu string kecuali data yang hendak dihapus. Setelah memilih data yang hendak dihapus, maka program akan mengikuti algoritma berikut:

1. Mencari letak data yang akan dihapus. Hal ini dilakukan dengan cara mencocokkan nama pemancar (komponen pertama dari data) dengan nama pemancar yang diisi pada *user interface*.
2. Memindahkan isi data pertama sampai data sebelum data yang akan dihapus ke dalam variabel string
3. Menambahkan isi data setelah data yang akan dihapus ke dalam variabel string yang sama.
4. Menulis variabel string tersebut ke dalam *file* yang sama. Gambar 4.5 memperlihatkan metode `deleteData( String namaFile, string namaData )` yang digunakan pada program ini.

```
public static void deleteData( String namaFile, String input )
{
      String fill = "";
      String data = "";
      String nama = "";
      String NAMAFILE = namaFile + ".har";
      try
      {
      File file=new File(NAMAFILE);//Mendapatkan input dari teks
field pertama
      FileReader reader=new FileReader(file);
      BufferedReader baca=new BufferedReader(reader);
      fill=baca.readLine();//Membaca isi file
      StringTokenizer token = new StringTokenizer(fill ,";");
      nama = token.nextToken();

      if ( nama.equals(input))
      {
            fill = baca.readLine();//Membaca isi file
            if (!(fill.equals("eof")))
                  data = fill;
            while (!(fill.equals("eof"))) //while
```

```
(!(fill.equals("eof")))
            {
                  fill = baca.readLine();//Membaca isi file
                  if(!(fill.equals("eof")))//if(!(fill.equals("eof")))
                  {
                        data = data  + "\n" + fill ;
                  }
            }
      }
      else
      {
            while( !(fill.equals( "eof" )))
            {
                  while ( !(nama.equals( input )))
                  {
                        if ( !(nama.equals( input )))
                        {
                          data = data + fill + "\n";
                          fill = baca.readLine();
                          StringTokenizer token1 = new
StringTokenizer(fill ,";");
                          nama = token1.nextToken();
                        }
                  }
                  fill = baca.readLine();
                  data = data + fill + "\n";
                  }}
File out = new File(NAMAFILE);//mengambil nama file yang akan
ditulis output
FileWriter fileOut = new FileWriter(out);
BufferedWriter fileWriteOut = new BufferedWriter(fileOut);
fileWriteOut.write(data + "\n" + "eof" );
fileWriteOut.close();//Menyimpan file yang telah ditulis

fill = "";
data = "";

File file2=new File(NAMAFILE);
FileReader reader2=new FileReader(file2);
BufferedReader baca2=new BufferedReader(reader2);
fill=baca2.readLine();//Membaca isi file
data = fill;
while (!(fill.equals(""))) /
{
      fill = baca2.readLine();//Membaca isi file
      if(!(fill.equals("")))//if(!(fill.equals("eof")))
      {
            data = data + "\n" + fill;
      }}
```

```
File out2 = new File(NAMAFILE);
FileWriter fileOut2 = new FileWriter(out2);
BufferedWriter fileWriteOut2 = new BufferedWriter(fileOut2);
fileWriteOut2.write(data + "\n" + "eof" );
fileWriteOut2.close();
}
catch (Exception e )
{
    JOptionPane.showMessageDialog( “error deleting file” );
}
}
```

Gambar 4.5 Kode `deleteData( String nama`*`File`*`, String input)`

Perintah `if ( nama.equals(input))`digunakan apabila data yang hendak dihapus merupakan data yang terletak pada baris pertama dari daftar data pada *file*. Terlihat pada kode diatas, program melakukan *looping* mencocokkan nama pemancar pada tiap baris dengan nama pemancar yang hendak dihapus, sampai “eof” ditemukan. Apabila nama pemancar tidak sesuai, maka program akan menambahkan satu baris data tersebut kedalam sebuah string. Apabila program menemukan nama pemancar dalam daftar datadi dalam *file* tersebut sama dengan nama pemancar yang hendak dihapus oleh *user*, maka program tidak menambahkan satu baris data tersebut ke dalam string, tetapi melewatkannya dan membaca baris berikutnya. Demikianlah dihasilkan sebuah string yang memiliki semua baris data kecuali baris data yang hendak dihapus.

Update data

Update data adalah sebuah proses dimana *user* dimampukan untuk mengubah data yang lama dengan data yang baru, yang dilakukan di dalam *file* yang sama. Pengubahan tersebut dapat berupa mengganti nama pemancar, nilai lintang, nilai bujur, dan daftar nomor-nomor kanal. Ide dasar dari proses *update* data adalah dengan menghapus data yang hendak diubah, kemudian menambahkan kepada *file* yang sama, data yang telah mengalami perubahan. Jadi logika proses *update* data pada program ini menggabungkan metode `deleteData` dan `addData`. Algoritma proses *update* data yang digunakan pada program optimalisasi kanal radio siaran FM ini adalah:

1. Menyimpan nama pemancar dari data lama.

2. Apabila data dengan nama pemancar seperti yang dimasukkan oleh *user* ditemukan, maka data tersebut akan dihapus dari *file*.
3. Menyimpan hasil perubahan yang dilakukan ke dalam bentuk string.
4. Menulis string tersebut ke dalam *file* yang sama. Hal ini berakibat data yang baru akan berada pada baris terakhir dari daftar data pemancar yang ada. Gambar 4.6 memperlihatkan sintaks kode untuk metode `updateData` pada program ini.

```
public static void updateData(String namaFile, String dataLama,
String dataBaru )
{
      String NAMAFILE = namaFile + ".har";
      deleteData( NAMAFILE, dataLama );
      addData( NAMAFILE, dataBaru );

}
```

Gambar 4.6. Kode untuk Metode `updateData( String nama`*`File`*`, String dataLama, String dataBaru)`

Metode `updateData` memiliki tiga parameter string, yaitu `namaFile`, `dataLama` dan `databaru`. `namaFile` merupakan nama propinsi tempat data tersebut disimpan, `dataLama` merupakan data yang hendak diubah, dan `dataBaru` merupakan hasil ubahan dari data lama.

OPTIMALISASI DISTRIBUSI KANAL

Optimalisasi yang dimaksud disini adalah sebuah proses pencarian nomor-nomor kanal, selain dari nomor-nomor yang sudah *exist*, yang masih dapat dialokasikan bagi suatu daerah tertentu dalam suatu propinsi. Nomor-nomor kanal baru tersebut merupakan nomor-nomor kanal yang bebas interferensi, baik terhadap pemancar-pemancar lain yang berdekatan, maupun dengan dirinya sendiri (*self-interference*). Dengan demikian, dengan mudah *user* dapat mengetahui apakah suatu daerah masih memiliki kemungkinan untuk menambah jatah nomor kanal berikut nomor-nomor kanal berapa saja yang bisa diberikan.

Input yang diminta program meliputi nama propinsi yang akan dikonsider, nama pemancar, spasi frekuensi antar pemancar yang hendak dioptimalisasi dengan pemancar yang dikonsider, serta nomor kanal terendah dari nomor-nomor kanal yang sudah *exist*.

Pemancar-pemancar yang akan dikonsider sebagai potensi interferensi adalah pemancar-pemancar yang berada dalam radius 200km dari pemancar yang akan dioptimalisasi. Nomor kanal yang sama *(co-channel*) dapat diterapkan untuk pemancar-pemancar yang berjarak lebih dari 200km, sehingga program tidak perlu mengkonsider pemancar-pemancar tersebut.

User dapat memasukkan lebih dari satu propinsi, dan program akan menggabungkan semua daftar pemancar dari masing-masing propinsi. Hal ini dilakukan terutama untuk kasus dimana pemancar yang akan dioptimalisasi berada pada perbatasan dengan propinsi lain. Ketika hal ini terjadi, maka akan ada pemancar dari propinsi lain yang juga berpotensi interferensi karena, sekalipun berada pada propinsi yang berbeda, tetapi masuk dalam radius 200km.

4.3.1. Menentukan Pemancar-pemancar dalam Radius 200 Kilometer

Kelas `PilihDataKonsider`, dengan metode `op1(String nama`*`File`*`, string dataSasaran)` digunakan untuk mencari pemancar-pemancar mana saja yang berjarak 200km dari pemancar yang akan dioptimalisasi. Perhitungan jarak pemancar satu dengan yang lain dilakukan dengan mengambil nilai lintang dan bujur kedua pemancar, dimana selisih setiap satu derajat adalah 111.11 km. Setelah nilai lintang dan bujur diubah ke desimal, maka perhitungan jarak antara dua pemancar dilakukan dengan menggunakan rumus mencari jarak dua titik, yaitu akar dari kuadrat selisih lintang dan bujur dua pemancar.

Ide dasar dari proses ini adalah pertama-tama dengan memindahkan beberapa komponen dari informasi yang dibutuhkan dari masing-masing pemancar ke dalam sebuah *array* multi dimensi. Dimensi dari *array* ini adalah n x 4, dimana n adalah jumlah baris, dan 4 adalah jumlah kolom. Jumlah baris tidak dapat ditentukan sebelumnya karena tergantung dari berapa banyak jumlah pemancar dalam *file* tersebut. Jumlah kolom dapat dibuat tetap dan berisi: nama pemancar, nilai lintang, nilai bujur, dan daftar nomor kanal. Gambar 4.7 memperlihatkan sintaks untuk memindahkan data pada *file* kedalam `array[n][4].`

```
try
{
String names[][] = new String [n][4];
int i = 0;
File file=new File(NAMAFILE);
FileReader reader=new FileReader(file);
BufferedReader baca=new BufferedReader(reader);
for ( int count = 0; count < n; count ++ )
{
        fill=baca.readLine();
        String nama = "";
        String LONG = "";
        String LAT = "";
        String napem = "";
        String tampung = "";
        StringTokenizer token = new StringTokenizer(fill ,";");
        while(token.hasMoreTokens())
        {
         nama = token.nextToken();
         LONG = token.nextToken();
         LAT = token.nextToken();
         String kanal1 = token.nextToken();
         tampung = kanal1;
         while (token.hasMoreTokens())
         {
           String kanal = token.nextToken();
           tampung = tampung + ";" + kanal;
         }
        }
        names[i][0] = nama;
        names[i][1] = LONG;
        names[i][2] = LAT;
        names[i][3] = tampung;
        i = i + 1;
}
```

Gambar 4.7 Sintaks Memindahkan Data ke dalam *array*

Dari kode diatas didapat sebuah *array* `names[n][4]` yang isinya merupakan komponen-kompenen dalam baris-baris data. Komponen-komponen pada baris data dipisahkan oleh tanda titik koma, sebagai penanda berakhirnya komponen yang satu satu masuk komponen berikutnya, seperti telah dijelaskan sebelumnya. Daftar kanal dikumpulkan ke dalam satu kolom yang nantinya akan dipecah-pecah lagi dan dikumpulkan ke dalam *array* yang lain.

Langkah berikutnya adalah dengan membuat sebuah *array* (NAMES[][]) baru dengan dimensi (n-1) x 4. Isi *array* ini adalah seluruh isi *array* names[][], kecuali baris dimana data yang

hendak dioptimalisasi berada. Proses selanjutnya adalah melakukan *looping* membandingkan nilai lintang dan bujur dari pemancar yang hendak dioptimalisasi dengan semua pemancar pada NAMES[][]. Gambar 4.8 memperlihatkan kode bagaimana proses ini dilakukan.

```
//cari data keberapa dari array yang jadi sasaran
                int dapat = 0;
                for ( int z = 0;  z < n; z++ )
                {
                     String cari = names[z][0];
                     if ( cari.equals( dataSasaran ))
                          dapat = z;
                }
//buat array yang isinya data selain sasaran
                String NAMES[][] = new String [n][4];
                for ( int zz = 0; zz < dapat; zz++ )
                {
                     NAMES[zz][0] = names[zz][0];
                     NAMES[zz][1] = names[zz][1];
                     NAMES[zz][2] = names[zz][2];
                     NAMES[zz][3] = names[zz][3];
                }
                for ( int zzz = dapat ; zzz < n-1 ; zzz++ )
                {
                     NAMES[zzz][0] = names[zzz+1][0];
                     NAMES[zzz][1] = names[zzz+1][1];
                     NAMES[zzz][2] = names[zzz+1][2];
                     NAMES[zzz][3] = names[zzz+1][3];
                }
```

```
double kuadradJarak = 0;
double jarak = 0;
int jmlKonsider = 0;
for ( int f = 0; f < n-1; f ++ )
{
   kuadradJarak = ((Double.parseDouble(names[dapat][1])) -
   (Double.parseDouble(
   NAMES[f][1])))*((Double.parseDouble(names[dapat][1])) -
   (Double.parseDouble( NAMES[f][1]))) +
   ((Double.parseDouble(names[dapat][2])) - (Double.parseDouble(
   NAMES[f][2])))*((Double.parseDouble(names[dapat][2])) -
   (Double.parseDouble( NAMES[f][2])));
   jarak = Math.sqrt(kuadradJarak);
   if ( jarak < RANGE )
      jmlKonsider = jmlKonsider + 1;
 }
        String dataKonsider[][] = new String[jmlKonsider][4];
        int v = 0;
        for ( int h = 0; h < n-1; h ++ )
        {
            kuadradJarak = ((Double.parseDouble(names[dapat][1])) -
            (Double.parseDouble(
            NAMES[h][1])))*((Double.parseDouble(names[dapat][1])) -
            (Double.parseDouble( NAMES[h][1]))) +
            ((Double.parseDouble(names[dapat][2])) -
            (Double.parseDouble(
            NAMES[h][2])))*((Double.parseDouble(names[dapat][2])) -
            (Double.parseDouble( NAMES[h][2])));
            double jarak2 = Math.sqrt(kuadradJarak);
            jarak = jarak2 * 111.111;
            if ( jarak < RANGE )
            {
               dataKonsider[v][0] = NAMES[h][0];
               dataKonsider[v][1] = NAMES[h][1];
               dataKonsider[v][2] = NAMES[h][2];
               dataKonsider[v][3] = NAMES[h][3];
               v= v + 1;
            }}
DATAKONSIDER = DATAKONSIDER +dataKonsider[0][0] + ";" +
dataKonsider[0][1] + ";" + dataKonsider[0][2] + ";" +
dataKonsider[0][3] ;
for ( int hh = 1; hh < jmlKonsider; hh ++ )
DATAKONSIDER = DATAKONSIDER + "\n" + dataKonsider[hh][0] + ";" +
dataKonsider[hh][1] + ";" + dataKonsider[hh][2] + ";" +
dataKonsider[hh][3];
DATAKONSIDER = DATAKONSIDER + "\neof";
File out = new File("daftarKonsider.har");//mengambil nama file yang
akan ditulis output
FileWriter fileOut = new FileWriter(out);
BufferedWriter fileWriteOut = new BufferedWriter(fileOut);
fileWriteOut.write(DATAKONSIDER);//Menulis isi dari variabel
penampung ke file
fileWriteOut.close();//Menyimpan file yang telah ditulis
```

Gambar 4.8 Mencari Data Pemancar Konsider

Proses *looping* dengan membandingkan nilai lintang dan bujur pemancar dilakukan dua kali, pertama untuk mengetahui berapa jumlah data yang akan dikonsider. Hal ini dilakukan sehingga sebuah *array* baru `daftarKonsider[]` dapat diinisialisasi. *Looping* kedua adalah untuk memasukkan data pemancar yang akan dikonsider kedalam `daftarKonsider[].` Bagian terakhir dari langkah-langkah memilih data pemancar yang dikonsider adalah memindahkan isi `daftarKonsider[]` ke dalam sebuah string dan menulisnya ke dalam *file* daftarKonsider.har. *File* ini yang nantinya akan diproses lebih lanjut.

4.3.2. Proses Optimalisasi

Proses optimalisasi dilakukan dengan kelas `OptimalisasiKanal` yang memiliki metode `hitungJumlahKanal(String baris)`, metode `bacaFile(String namaFile)`, dan metode `optimalisasiFM(String sumber, String fill, int spasi, int jmlKanal)`. Ketiga metode ini akan dipanggil oleh metode `op2(int jmlPemancar, int nomorKanalTerkecil)` dari kelas `OpInterface`.

Logika dari proses ini dikerjakan oleh metode `optimalisasiFM` dimana sebuah sumber nomor-nomor kanal diproses terhadap sebuah daftar nomor kanal yang lain, dan dicari nomor-nomor kanal berapa saja dari sumber tersebut yang tidak interferensi dengan nomor-nomor kanal pada daftar itu. Spasi frekuensi diperlukan di dalam proses ini untuk mengetahui jarak aman bebas interferensi antara dua pemancar. Gambar 4.9 memperlihatkan sintaks kode untuk metode `optimalisasiFM`.

Nomor-nomor kanal awal adalah sebuah *array* yang memiliki 204 komponen yang berisi nomor-nomor kanal dari 1 sampai 201. Hal ini dikemukakan dalam Keputusan Menteri Perhubungan no. 15 tahun 2003 tentang Rencana Induk (*Master Plan*) Frekuensi Radio Penyelenggaraan Telekomunikasi Khusus Untuk Keperluan Radio Siaran FM (*Frequency Modulation*), dimana rentang frekuensi radio siaran FM adalah dari 87.6MHz sampai 107.9MHz, dengan total 204 nomor kanal masing-masing dengan *bandwidth* 0.1MHz. Nomor kanal 202 – 204 diperuntukkan bagi radio komunitas (pasal 5 ayat2b).

```
public static String optimalisasiFM( String sumber, String
fill, int spasi, int jmlKanal )
{

String pemancar[] = new String[jmlKanal + 1];

int bin1[] = new int[201];
int bin2[] = new int[201];
int d = 0;
int n = -1;
int a = 0;
int aa = 0;


String napem = "";
String kanal = "";

StringTokenizer token1 = new StringTokenizer(sumber ,";");
while (token1.hasMoreTokens())
{

      String q = token1.nextToken();
      //System.out.println( q );
      bin1[aa] = Integer.parseInt(q);
      while (token1.hasMoreTokens())
      {
            String kanal2 = token1.nextToken();
            aa = aa +1;
            bin1[aa] = Integer.parseInt(kanal2);
      }
}

StringTokenizer token = new StringTokenizer(fill ,";");
while (token.hasMoreTokens())
{
      napem = token.nextToken();
      String LONG = token.nextToken();
      String LAT = token.nextToken();
      kanal = token.nextToken();
      pemancar [a] = napem;
      a = a + 1;
      pemancar[a] = kanal;
      while (token.hasMoreTokens())
      {
            a = a +1;
            String kanal1 = token.nextToken();
            pemancar[a] = kanal1;
      }
      }
```

```
nextRow1:
     for ( int counterA = 0; counterA < bin1.length;
counterA++)
     {
          nextRowA:
          for (int channelA = 1; channelA < pemancar.length;
channelA++)
          {
           d= bin1[counterA] -
Integer.parseInt(pemancar[channelA]);
          //ambil nilai mutlak a
          if (d < spasi)
          d = d*(-1);
          if ( bin1[counterA] == 0)
              break;
          //logika optimalisasi
          if ( d < spasi )
               continue nextRow1;
          else
              if (channelA == pemancar.length - 1)
             {
               n += 1;
               //output+=counterA+" ";
               bin2[n] = bin1[counterA];
            }
            else
               continue nextRowA;
          }//end of for channelA

}//end of for counterA

for ( int c = 0; c < bin1.length; c++)
{
    bin1[c] = bin2[c];
    bin2[c] = 0;
}
String xx = "";
for ( int z = 0; z < bin1.length; z ++)
xx = xx + String.valueOf(bin1[z]) + ";";
return xx;
}
```

Gambar 4.9 Metode OptimalisasiFM

Metode `optimalisasiKanal` berfungsi untuk mencari nomor-nomor kanal yang tidak mengalami inteferensi dengan kanal-kanal yang sudah dialokasikan untuk suatu pemancar.

Nomor-nomor kanal awal merupakan variabel string sehingga perlu dipecah-pecah dan dimasukkan ke dalam *array* `bin1[201]`. Seperti dijelaskan sebelumnya, nomor-nomor kanal awal adalah daftar nomor kanal dari 1 sampai 201. Selanjutnya kanal-kanal dalam pemancar tersebut juga dimasukkan ke dalam *array* `pemancar[].` *Array* ini memiliki komponen sebanyak jumlah kanal + 1. Perlu ditambah 1 karena komponen pertama *array* ini adalah nama pemancar yang bersangkutan.

Setelah dihasilkan kedua *array* `bin1[]` dan pemancar[], maka selanjutnya program akan mencari nomort-nomor kanal dalam `bin1[]` yang tidak interferensi dengan nomor-nomor kanal pada `pemancar[].` Nomor-nomor kanal ini kemudian dimasukkan kedalam `bin2[],` yang berfungsi sebagai tempat penyimpanan sementara. Setelah `bin2[]` selesai diisi, maka seluruh isi komponen `bin1[]` akan diganti dengan isi komponen `bin2[],` dan `bin1[]` sekarang kembali lagi menjadi nomor-nomor kanal yang tersedia.

Metode `op2` pada kelas `OpInterface` melakukan *looping* untuk mencari nomor-nomor kanal dari sumber yang bebas interferensi sebanyak jumlah pemancar yang dikonsider, sehingga dihasilkan nomor-nomor kanal yang bebas interferensi dengan semua nomor-nomor kanal pemancar yang dikonsider, sesuai dengan spasi frekuensi. Gambar 4.10 memperlihatkan bagian dari kode metode `op2` yang memanggil metode `hitungJumlahKanal`.

```
File file=new File("daftarKonsider.har");//Mendapatkan input
dari teks field pertama
          FileReader reader=new FileReader(file);
          BufferedReader baca=new BufferedReader(reader);
          opKanal = new OptimalisasiKanal();
          for ( int i = 0; i < jmlPemancar; i++)
          {
               fill = baca.readLine();//Membaca isi file
               int b = opKanal.hitungJumlahKanal(fill);
               sumber = opKanal.optimalisasiFM( sumber, fill,
spasi[i], b );
          }
```

Gambar 4.10 Looping Metode `OptimalisasiKanal`

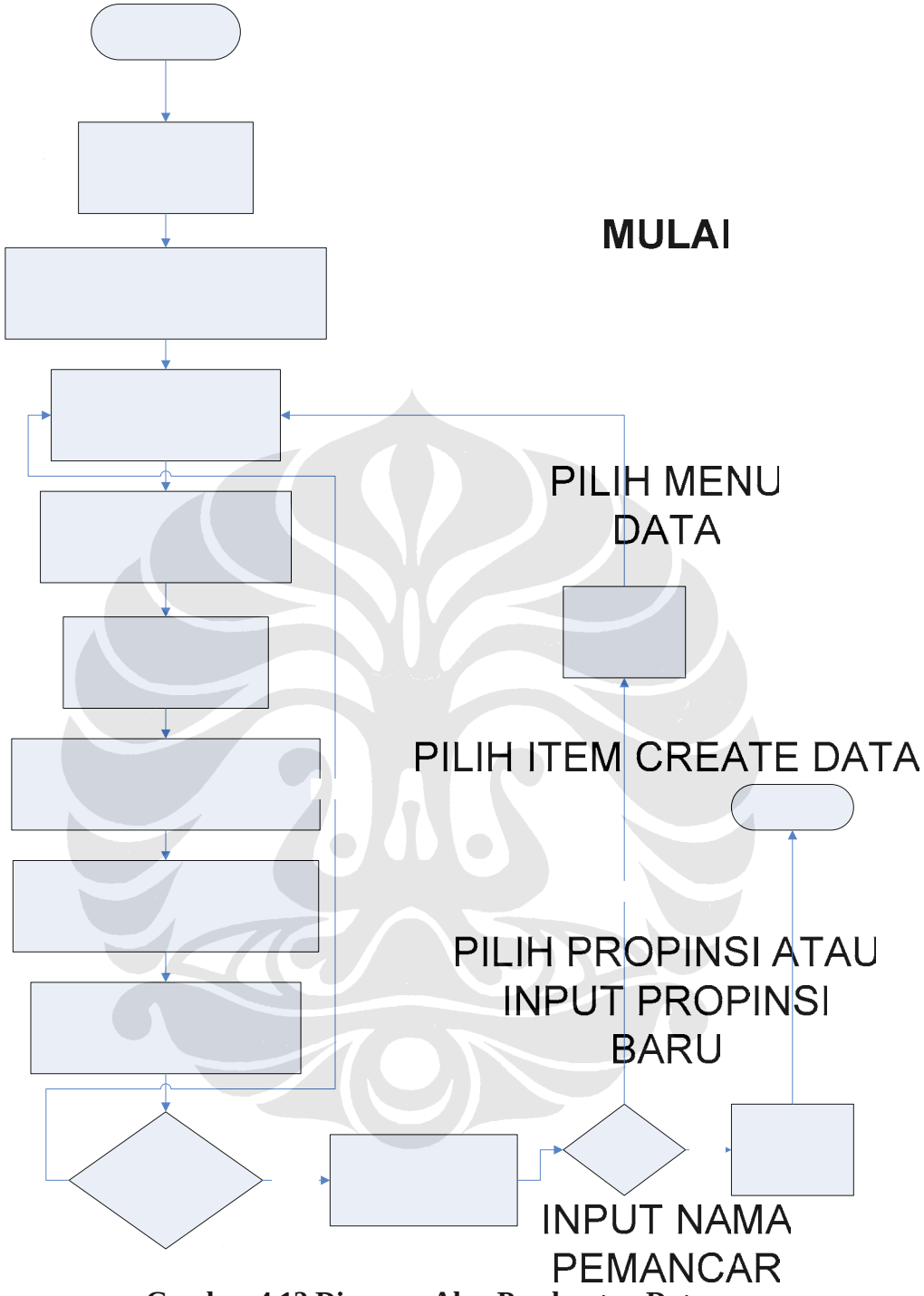
Bagian terakhir adalah dengan membuang dari semua nomor-nomor kanal yang sudah dihasilkan, nomor-nomor yang mengalami interferensi dengan pemancar tu sendiri (*self-interference*). Nomor-nomor kanal akan bebas dari *self-interference* ketika jarak antara nomor yang satu dengan yang lain minimal 4 (*adjacent* 4). Oleh karena itu, dilakukan looping antara sesame nomor kanal, mencari nomor-nomor kanal yang tidak mengalami *self-interference*, dan mengumpulkannya ke dalam sebuah string. Gambar 4.11 memperlihatkan bagian kode dari metode op2 yang mengerjakan proses ini.

```
//harus bebas self interference
          String sumber3 = String.valueOf(kanal2[0]);
          int w = 1;
          int j = 0;
          int jj = 1;
          while ( w != 0)
          {
               int gu = Math.abs(kanal2[j] - kanal2[jj]);
               if ( gu >= 4 )
               {
                    sumber3 = sumber3+ ";" + kanal2[jj];
                    j = jj;
                    jj = jj + 1;
               }
               else
                    jj = jj + 1;
               if ( jj == hitung-1)
               {
                    gu = Math.abs(kanal2[j] -
kanal2[jj]);
                    if ( gu >= 4 )
                    {
                         sumber3 = sumber3+ ";" +
kanal2[jj];
                         j = jj;
                         jj = jj + 1;
                    }
                    w = 0;
               }
          }
          hasilArea.setText ( sumber3);
```

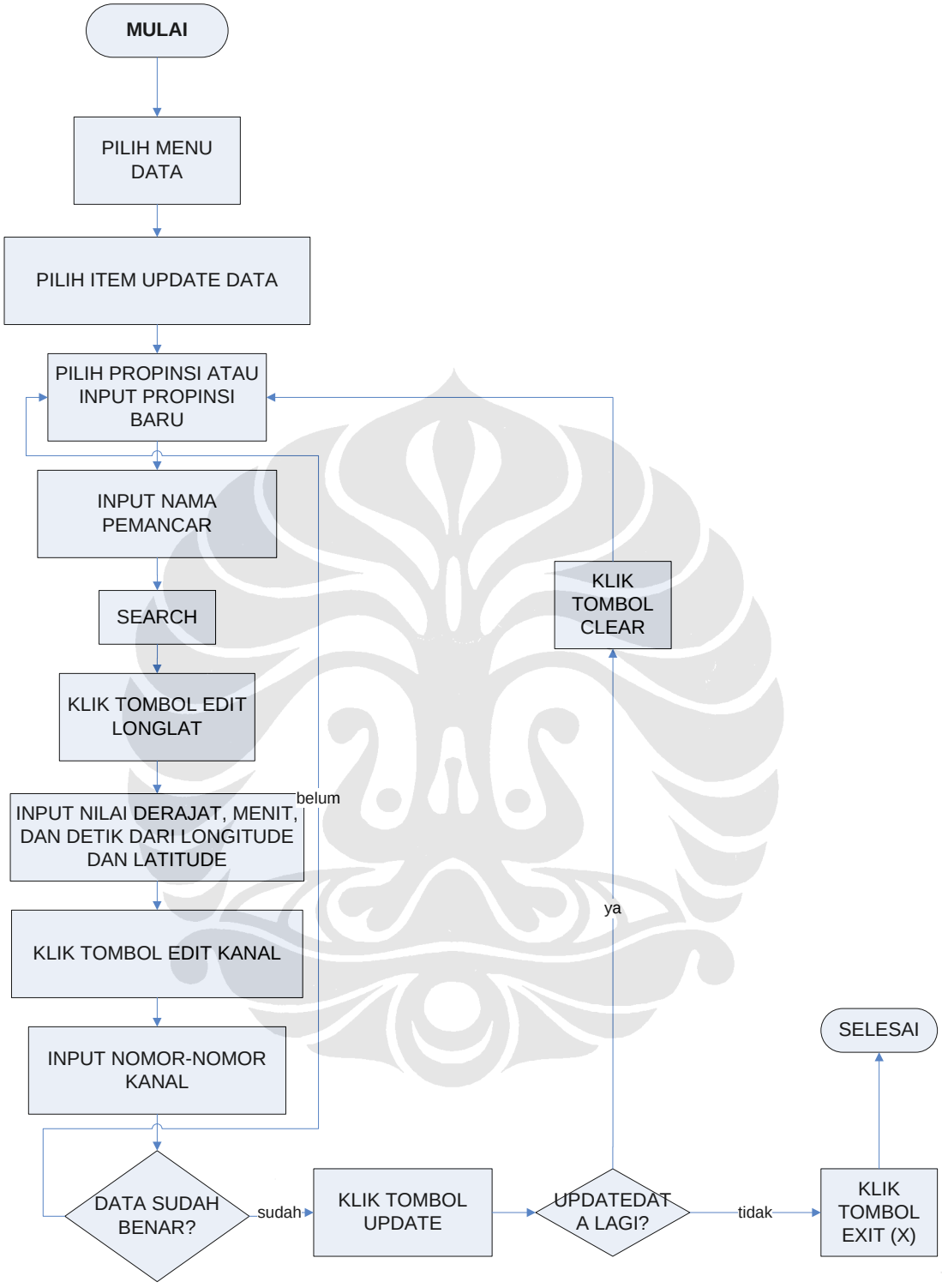
Gambar 4.11 Proses Bebas *self-interference*

4.4 DIAGRAM ALUR PENGGUNAAN PROGRAM OLEH *USER*

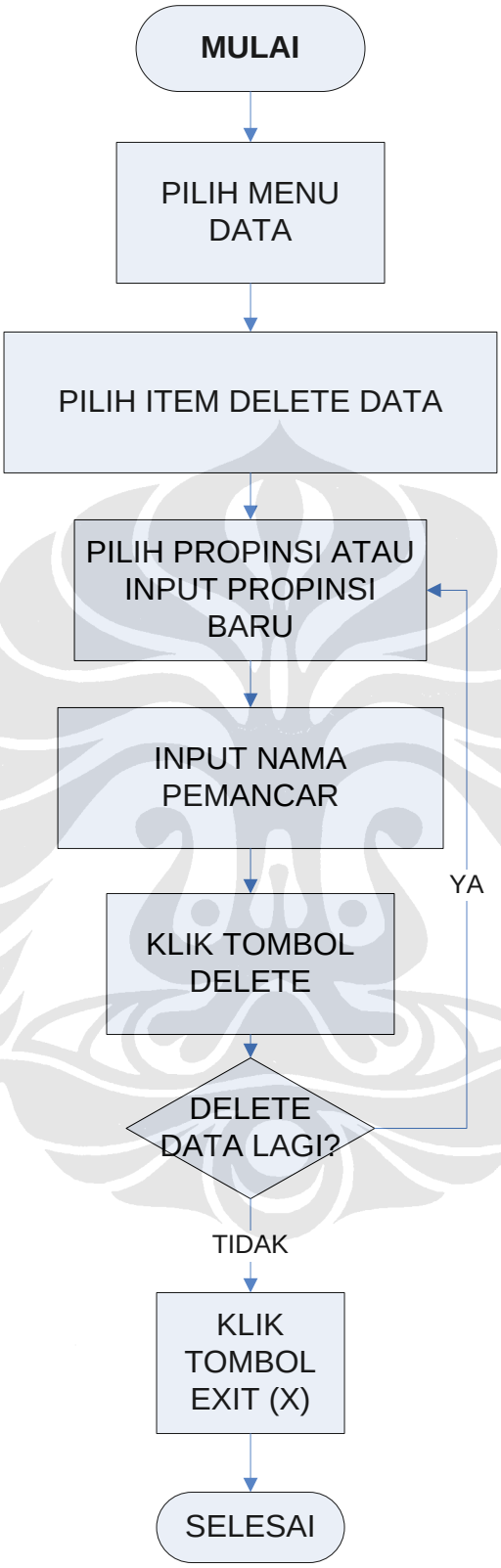
Gambar 4.12 sampai 4.15 memperlihatkan diagram alur bagaimana *user* menggunakan program optimalisasi kanal radio siaran FM ini.



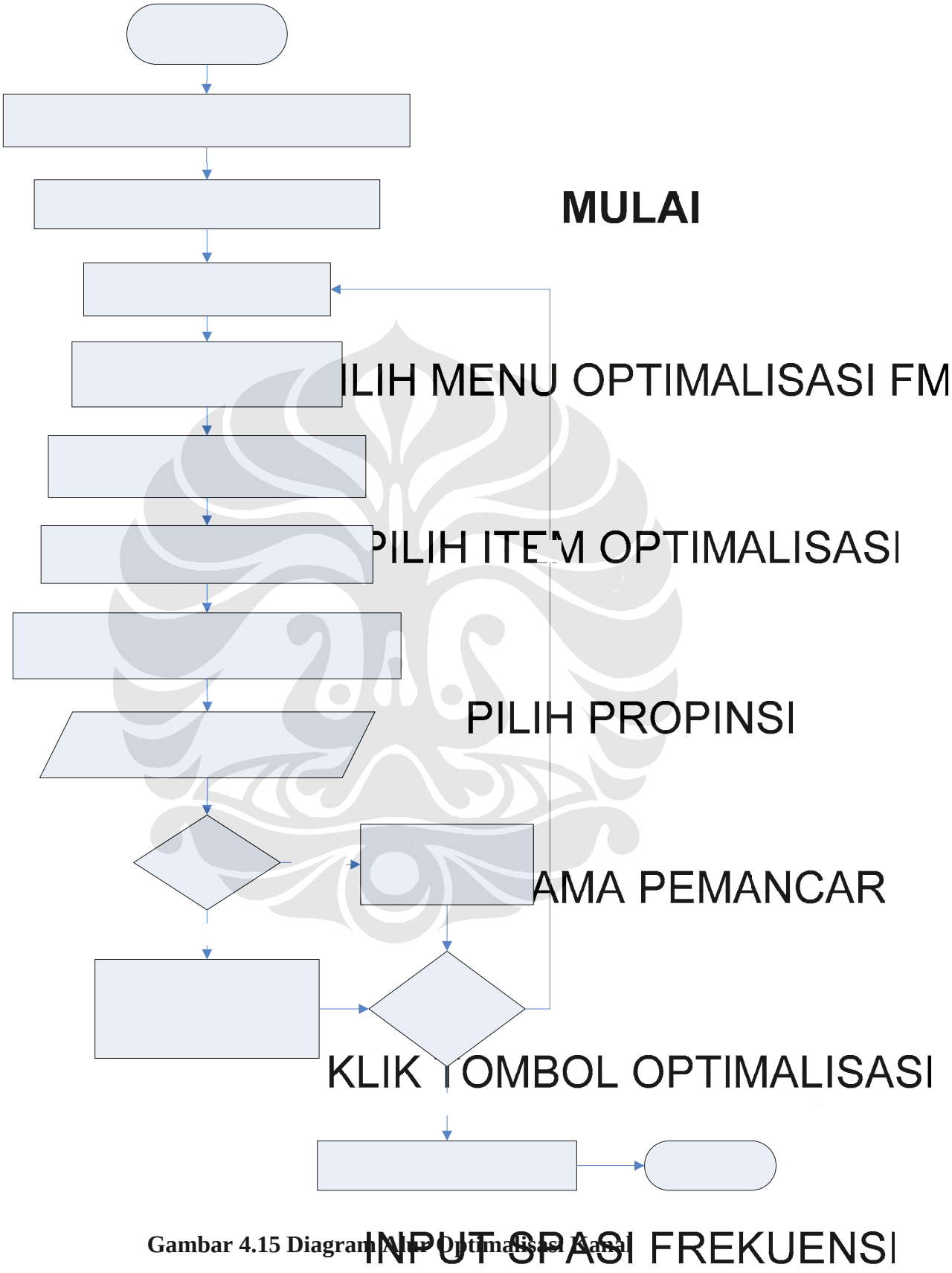
Gambar 4.12 Diagram Alur Pembuatan Data



Gambar 4.13 Diagram Alur *update* Data



Gambar 4.14 Diagram Alur Penghapusan Data



Gambar 4.15 Diagram Alur Optimalisasi Kanal

INPUT NOMOR KANAL TERKECIL

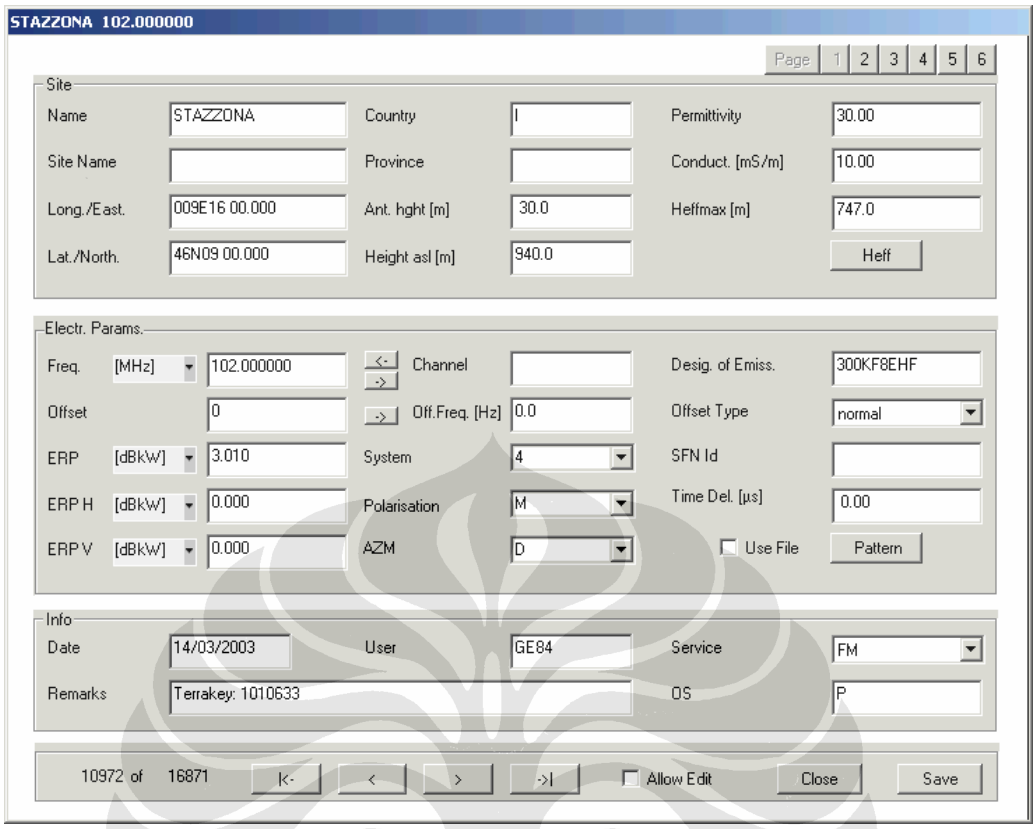
BAB V
PERHITUNGAN SPASI FREKUENSI

5.1 *CONTOUR CALCULATION*

Perhitungan kontur dilakukan untuk mengatur wilayah layanan (*service area*) suatu pemancar sedemikian sehingga sesuai dengan ketetapan yang sudah ditentukan mengenai radius maksimum pemancar. Perhitungan ini dilakukan dengan menggunakan *software* Chirplus BC.

Berikut langkah-langkah yang dikerjakan untuk melakukan simulasi kontur dengan menggunakan Chirplus BC:

1. Membuat pemancar baru pada *Working Database*
2. Memasukkan nama pemancar, nilai lintang dan bujur, dan tinggi antena pemancar. Tinggi antena pemancar disesuaikan dengan ketentuan dari Keputussan Menteri 15 tahun 2003, berdasarkan kelas pemancar dengan ketentuan seperti berikut:
   Kelas A ➔ 150 m
   Kelas B ➔ 80 m
   Kelas C ➔ 40 m
3. Perhitungan *Effective High Above Average Terrain* (EHAAT). Peta yang digunakan dalam perhitungan EHAAT adalah DTM 200.
4. Masukan frekuensi diset 100MHz
5. Polarisasi di-*set* vertikal
6. Penentuan nilai ERP awal 1 KWatt
7. Simpan konfigurasi pemancar. Gambar 5.1 memeprlihatkan tampilan konfigurasi suatu pemancar.



Gambar 5.1 Tampilan Konfigurasi Pemancar pada Chirplus BC

8. Perhitungan kontur dilakukan dengan ketentuan berikut:
   - Mode kalkulasi yang digunakan adalah ITU-R 1456 *Database*
   - Tinggi pemancar penerima (*Rx Height*) adalah 10 m.
   - *Show Text Window* diberi *check list*.

     Text Window merupakan jendela yang berisi informasi radius yang dijangkau oleh pemancar.
   - Jenis peta yang digunakan adalah DTM 200-Indonesia
   - *Field Strength* minimum adalah 66dBµV/m.

     Nilai ini di-*set* sama untuk semua kelas, sekalipun sebelumnya pada dasar teori disebutkan bahwa kelas yang berbeda memiliki karakteristik *field strength* yang berbeda pula.
   - *Save* parameter jika radius jangkauan layanan telah memenuhi spesifikasi.

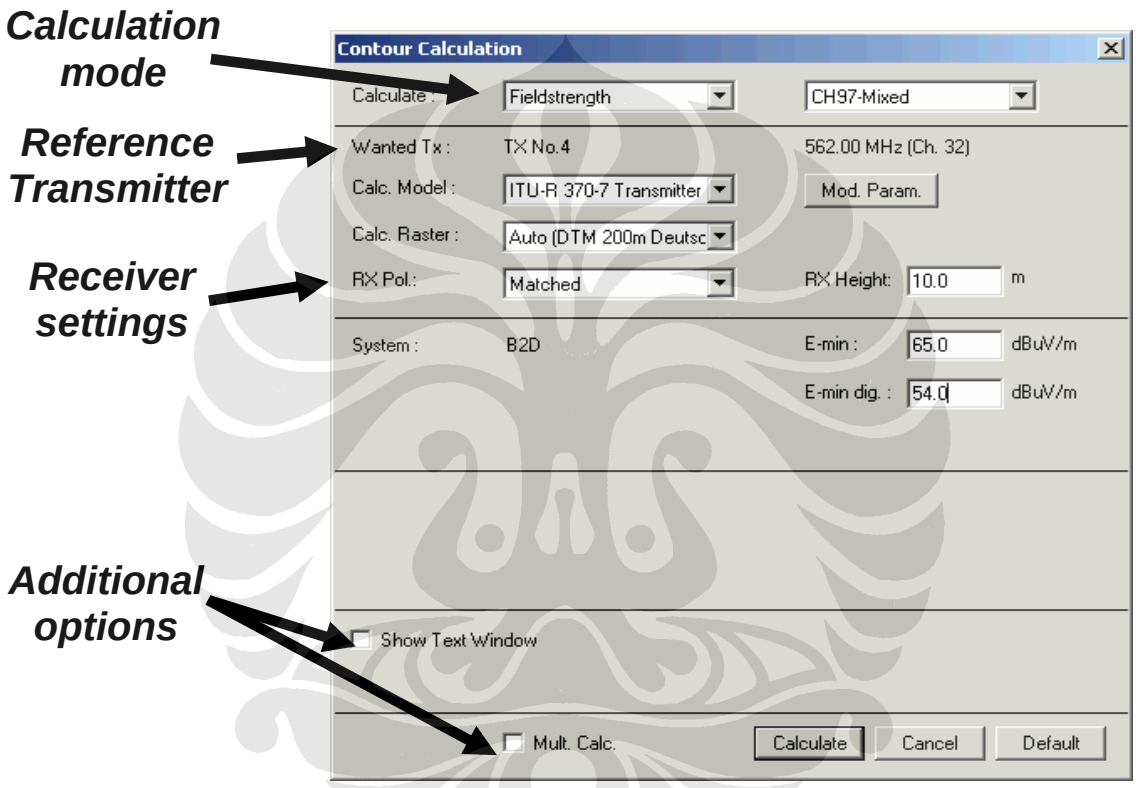
9. Pengaturan nilai ERP, sedemikian sehingga jarak maksimum kontur sesuai dengan Keputusan Menteri no. 15 tahun 2003, yaitu:

   Kelas A ➔ 30-31 kilometer

   Kelas B ➔ 20-21 kilometer

   Kelas C ➔ 12-13 kilometer

   Gambar 5.2 memperlihatkan tampilan perhitungan kontur Chirplus BC.



Gambar 5.2 Tampilan *Contour Caculation* Chirplus BC

Hasil pehitungan kontur seluruh pemancar pada wilayah Daerah Istimewa Yogyakarta dapat dilihat pada Lampiran A1.

5.2 PENENTUAN PEMANCAR BERPOTENSI INTERFERENSI

Untuk menentukan spasi frekuensi pada pemancar-pemancar pada suatu propinsi, ada kemungkinan dimana pemancar dari propinsi lain ikut menjadi potensi interferensi karena jarak wilayahnyayang berdekatan. Untuk itu, perlu dilakukan proses pencarian daerah potensi interferensi tersebut.

Proses pencarian daerah potensi interferensi juga dilakukan untuk efisiensi dalam proses perhitungan spasi optimalisasi kanal radio siaran FM, karena tidak perlu mencari spasi frekuensi untuk semua pemancar dalam radius 200 kilometer dari pemancar *wanted*. Daerah-daerah yang bukan potensi interferensi dapat langsung *co-channel* dengan pemancar *wanted*.

Proses ini dilakukan secara manual dengan menggunakan Chirplus BC, dimana semua kontur pemancar lain yang memasuki wilayah kontur pemancar *wanted* menjadi potensi interferensi, sedangkan kontur yang terpisah dengan kontur pemancar *wanted* bukanlah potensi interferensi bagi pemancar *wanted* tersebut.

Untuk setiap pemancar dilakukan dua kali pencarian daerah potensi interferensi, pertama pemancar yang dituju diset *wanted* dan pemancar lain disekitarnya diset *unwanted*, kedua, pemancar yang dituju diset *unwanted* sedangkan semua pemancar disekitarnya diset *wanted*. Kedua pencarian ini akan menghasilkan daerah-daerah potensi interferensi yang berbeda. Daerah-daerah potensi interferensi total merupakan logika OR dari semua potensi interferensi yang telah dicari.

Ketika pemancar diset *wanted*, maka ketika simulasi kontur dilakukan, nilai Emin diset 66dBμV, sedangkan ketika pemancar di-*set unwanted*, nilai Emin di-*set* 21dBμV.

Tabel 5.1 memperlihatkan daftar daerah potensi interferensi bagi masing-masing pemancar dalam propinsi Daerah Istimewa Yogyakarta yang telah dikerjakan.

Tabel 5.1 Daerah Potensi Interferensi bagi Masing-masing Pemancar dalam Propinsi Daerah Istimewa Yogyakarta

Wanted	*unwanted*	*unwanted*	*wanted*
KOTA	Karangkobar	KOTA	Banjarnegara

YOGYAKARTA		YOGYAKARTA	
	Kebuman		KaliBening
	Ambal		Karangkobar
	Bruno		Kebuman
	Kutoarjo		Sempor
	Klaten		Sadang
	Mungkit		Ambal
	Sukoharjo		Purworejo
	Grogol		Kutoarjo
	Griwoyo		Bruno
	Tirtomoyo		Leksono
	Ngadirojo		Kaliwiro
	Jatisruno		Sapuran
	Karang Anyar		Mungkid
	Karang Pandar		Boyolali
	Purwodadi		Klaten
	Sragen		Sukoharjo
	Jumantoro		Grogol
	Temanggung		Wonogiri
	Demak		Eromoko
	Sapuran		Griwoyo
	Kota Surakarta		Tirtomoyo
	Magelang		Ngadirojo
			Jatisruno
			Karang Anyar
			Karang Pandar
			Jumantoro

			Sragen
			Temanggung
			Magelang
			Sapuran
			Kota Surakarta
WATES, SENTOLO	Kalibening	WATES, SENTOLO	Ambal
	KARANG KOBAR		purworejo
	KEBUMEN		Kutoarjo
	Sempor		Mungkid
	Ambal		Klaten
	Purworejo		Eromoko
	Sadang		Magelang
	Ambal		
	Purworejo		
	Kutoarjo		
	Bruno		
	Wonosobo		
	Kaliwiro		
	Sapuran		
	Mungkid		
	Boyolali		
	Klaten		
	Sukoharjo		
	Grogol		

	Wonogiri		
	Eromoko		
	Griwoyo		
	Tirtomoyo		
	Ngadirojo		
	Jatisruno		
	Karang Anyar		
	Karang Pandar		
	Jumantoro		
	Sragen		
	Temanggung		
	Magelang		
	Kota Surakarta		
	Sapuran		
BANTUL, BANGUNTAPAN	Kebuman	BANTUL, BANGUNTAPAN	Ambal
	Ambal		PURWOREJO
	Kutoarjo		KUTOARJO
	KUTOARJO		BRUNO
	BRUNO		MUNGKID
	Mungkid		BOYOLALI
	Klaten		KLATEN
	SUKOHARJO		SUKOHARJO
	GROGOL		GROGOL
	GIRIWOYO		Jumantoro
	TIRTOMOYO		Karang Anyar

	NGADIROJO		Sapuran
	JATISRUNO		Magelang
	KARANG ANYAR		Kota Surakarta
	KARANG PANDAN		
	JUMANTONO		
	SRAGEN		
	Temanggung		
	Magelang		
	Kota Surakarta		
	Sapuran		
WONOSARI	Ambal	WONOSARI	AMBAL
	Bruno		PURWOREJO
	Kutoarjo		KUTOARJO
	Mungkid		BRUNO
	Boyolali		MUNGKID
	KLATEN		KLATEN
	SUKOHARJO		SUKOHARJO
	GROGOL		GROGOL
	EROMOKO		WONOGIRI
	GIRIWOYO		EROMOKO
	TIRTOMOYO		GIRIWOYO
	NGADIROJO		TIRTOMOYO
	JATISRUNO		NGADIROJO
	KARANG		JATISRUNO

	ANYAR		
	KARANG PANDAN		KARANG ANYAR
	SRAGEN		KARANG PANDAN
	TEMANGGUNG		SRAGEN
	KOTA SURAKARTA		JUMANTONO
	KOTA MAGELANG		KOTA MAGELANG
	Sapuran		TEMANGGUNG
			KOTA SURAKARTA
			Sapuran
PATOK	AMBAL	PATOK	KEBUMEN
	BRUNO		SADANG
	KUTOARJO		AMBAL
	KEBUMEN		PURWOREJO
	BOYOLALI		KUTOARJO
	MUNGKID		BRUNO
	KLATEN		LEKSONO
	SUKOHARJO		KALIWIRO
	GROGOL		SAPURAN
	NGADIROJO		MUNGKID
	TIRTOMOYO		BOYOLALI
	GIRIWOYO		KLATEN
	JATISRUNO		SUKOHARJO

	KARANG ANYAR		GROGOL
	KARANG PANDAN		WONOGIRI
	JUMANTONO		EROMOKO
	SRAGEN		GIRIWOYO
	TEMANGGUNG		TIRTOMOYO
	DEMAK		NGADIROJO
	KOTA SURAKARTA		KARANG ANYAR
	KOTA MAGELANG		KARANG PANDAN
	SAPURAN		JUMANTONO
			SRAGEN
			PURWODADI
			KOTA MAGELANG
			KOTA SALATIGA
			KOTA SURAKARTA
			TEMANGGUNG
			UNGARAN
			AMBARAWA
SLEMAN, NGAGLIK, KALASAN, NGEMPLAK, PAKEM	KEBUMEN	SLEMAN, NGAGLIK, KALASAN, NGEMPLAK, PAKEM	AMBAL

	BRUNO		KEBUMEN
	LEKSONO		KALIWIRO
	AMBAL		SAPURAN
	MUNGKID		PURWOREJO
	KLATEN		KUTOARJO
	SUKOHARJO		BRUNO
	GROGOL		MUNGKID
	GIRIWOYO		KLATEN
	TIRTOMOYO		SUKOHARJO
	NGADIROJO		GROGOL
	JATISRUNO		WONOGIRI
	PURWODADI		EROMOKO
	WIROSARI		GIRIWOYO
	DEMAK		TIRTOMOYO
	TEMANGGUNG		NGADIROJO
	SAPURAN		JATISRUNO
	KOTA MAGELANG		KARANG ANYAR
	KOTA SURAKARTA		KARANG PANDAN
			JUMANTONO
			KOTA MAGELANG
			KOTA SURAKARTA
GAMPING	KEBUMEN	GAMPING	KEBUMEN

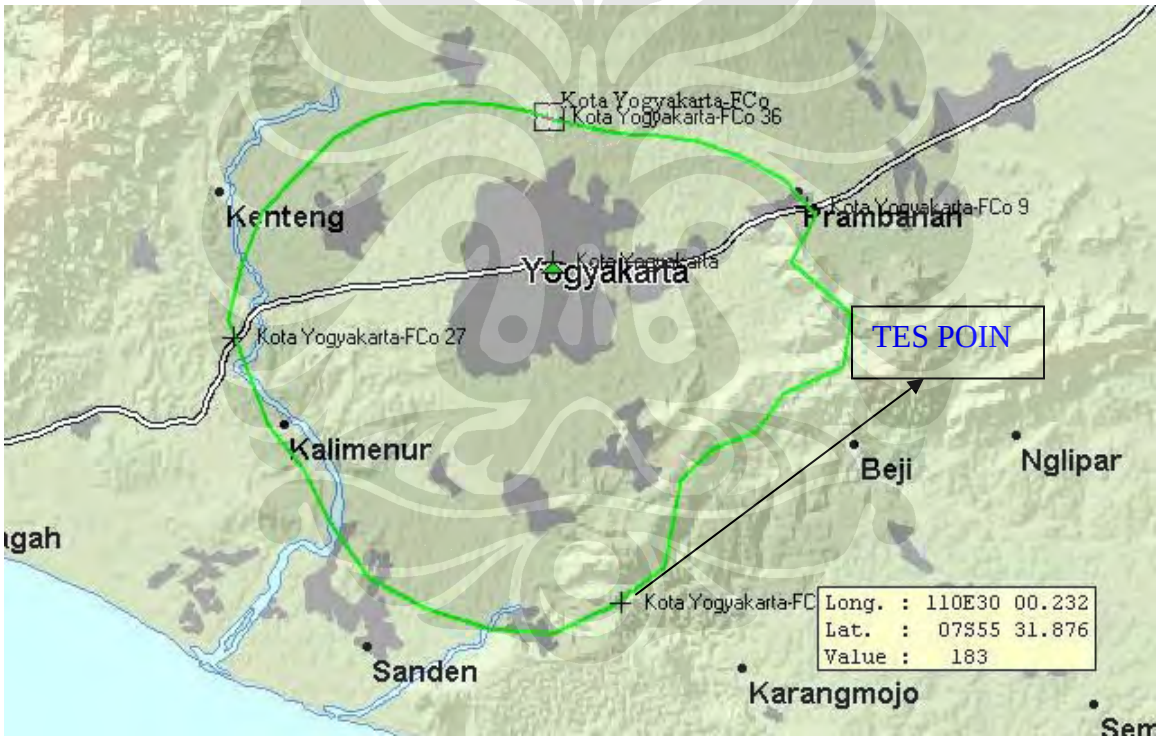
	KARANG KOBAR		SEMPOR
	AMBAL		SADANG
	KUTOARJO		AMBAL
	BRUNO		PURWOREJO
	MUNGKID		KUTOARJO
	KLATEN		BRUNO
	SUKOHARJO		LEKSONO
	GROGOL		KALIWIRO
	GIRIWOYO		SAPURAN
	TIRTOMOYO		MUNGKID
	JATISRUNO		KLATEN
	KARANG ANYAR		NGADIROJO
	KARANG PANDAN		JUMANTONO
	JUMANTONO		KOTA MAGELANG
	SRAGEN		
	DEMAK		
	TEMANGGUNG		
	SAPURAN		
	KOTA MAGELANG		
	KOTA SURAKARTA		
DEPOK	KUTOARJO	DEPOK	KEBUMEN

	AMBAL		AMBAL
	BRUNO		PURWOREJO
	MUNGKID		KUTOARJO
	KLATEN		BRUNO
	SUKOHARJO		MUNGKID
	GROGOL		KLATEN
	GIRIWOYO		SUKOHARJO
	TIRTOMOYO		GROGOL
	JATISRUNO		WONOGIRI
	KARANG ANYAR		EROMOKO
	KARANG PANDAN		GIRIWOYO
	JUMANTONO		NGADIROJO
	SRAGEN		JATISRUNO
	DEMAK		KARANG ANYAR
	SAPURAN		KARANG PANDAN
	TEMANGGUNG		JUMANTONO
	KOTA MAGELANG		KOTA MAGELANG
	KOTA SURAKARTA		KOTA SURAKARTA

5.3 *TEST POINT CALCULATION*

Perhitungan *test point* dilakukan untuk menghitung kuat medan minimum yang diterima oleh suatu pemancar dari pemancar lain. Kuat medan yang didapat dari perhitungan ini akan digunakan untuk mencari nilai *protection ratio* antara dua pemancar.

Pemberian tes poin dilakukan dengan menggunakan *vector test point* yang disediakan oleh Chirplus BC. Pemancar yang hendak diberi tes poin diaktifkan, kemudian pilih *vector test point* disebelah kanan bawah pada *panel*. Pada test point *database* kemudian akan muncul 36 tes poin yang mengelilingi batas kontur pemancar tersebut. Pilih 4 tes poin yang akan digunakan, yaitu tes poin ke 9, 18, 27, dan 36. Gambar 5.3 memperlihatkan simulasi kontur dan 4 test point pada pemancar Kota Yogyakarta. Gambar simulasi pemancar lainnya dapat dilihat pada lampiran B1.



Gambar 5.3 Simulasi kontur dan tes poin Kota Yogyakarta

Perhitungan kuat medan pada tiap-tiap pemancar dilakukan dengan menggunakan Chirplus BC. Seluruh tes poin diaktifkan, kemudian pilih *online calculation*. *Text Window* diaktifkan sehingga hasilnya dapat disimpan dalam bentuk *text*. Gambar cuplikan hasil perhitungan tes poin untuk pemancar Kota Yoyakarta

diperlihatkan oleh gambar 5.4. Hasil perhitungan tes poin lainnya dapat dilihat pada Lampiran B2.

General Data

Program Version 4.4.2
Calculated 16/04/08 23:25
User Name sysadmin
Calculation type Testpoint
Filename TP_W_TP_Vektor.txt

Transmitter Data

Name Kota Yogyakarta
Frequency [MHz] 100.000
Service FM
Oper. Status
Country INS
System 4
Coordinates 110E23 12.757 / 07S46 59.136
ERP [dBW] 34.3136
ERPV [dBW] 34.3136
Height (amsl) [m] 132.0
Antenna Height [m] 80.0
Polarisation V

Fieldstrength Calculation Settings

Model ITU-R 1546 Database
Receiver Height [m] 10.0
Receiver Polarisation Matched
Calc. Mode *Wanted*
Morpho Considered Yes
Topo Map no topo used
Morpho Map no morpho used

Time Prob. Steady [%] 50
Loc. Prob. Steady [%] 50
Use Land-Sea Disc. Yes
Use Cl.Ang. of Neg. Heff Yes
Adj. for diff. clim. Reg.not in use
Short urb. suburb. corr. in use

Point(s)	Dist(km)	Name	Longitude	Latitude	FST[dBµV/m]
TP 1:	11.610	Depok 1	110E29 21	07S45 48	70.2
TP 2:	10.692	Depok 2	110E23 58	07S52 40	75.0
TP 3:	6.648	Depok 3	110E19 39	07S47 24	79.9
TP 4:	6.789	Depok 4	110E24 24	07S43 32	72.2
TP 5:	3.106	Gamping 1	110E24 53	07S47 05	88.8
TP 6:	12.426	Gamping 2	110E21 28	07S53 26	73.8
TP 7:	15.841	Gamping 3	110E14 44	07S47 59	68.1
TP 8:	8.654	Gamping 4	110E19 46	07S43 51	73.4

Gambar 5.4 Cuplikan Hasil *text* Perhitungan Tes Poin

5.4 *PROTECTION RATIO* DAN SPASI FREKUENSI

Nilai *protection ratio* adalah selisih kuat medan dari pemancar *wanted* dengan kuat medan pemancar *unwanted*. Untuk setiap dua buah pemancar, dilakukan dua kali pehitungan nilai *protection ratio*, yaitu pemancar pertama diset *wanted* dan pemancar kedua diset *unwanted*, dan sebaliknya. Tabel 5.2 memperlihatkan perhitungan *protection ratio* pemancar Kota Yogyakarta dan pemancar Wates dimana pemancar Kota Yogyakarta diset sebagai *wanted*. Tabel 5.3 memperlihatkan memperlihatkan perhitungan *protection ratio* pemancar Kota Yogyakarta dan pemancar Wates dimana pemancar wates diset sebagai *wanted*

.

Tabel 5.2 Perhitungan *protection ratio* Pemancar Kota Yogyakarta(*wanted*) dan Pemancar Wates

wanted		*unwanted*	
Kota Yogyakarta		Wates, Sentolo	
66	TP1	68.1	
65.8	TP2	57.5	
65.8	TP3	53.1	
65.3	TP4	66.4	
		-2.1	adj3
		8.3	adj2
		12.7	adj2
		-1.1	adj2

Tabel 5.3 Perhitungan *protection ratio* Pemancar Kota Yogyakarta(*unwanted*) dan Pemancar Wates

Wanted		*unwanted*	
Wates, Sentolo		Kota Yogyakarta	
65.6	TP1	21.4	
65.9	TP2	45	
64.9	TP3	70	
67.6	TP4	26.6	

Kolom *Protection ratio*

Kolom *Protection ratio*

		44.2	adj1
		20.9	adj2
		-5.1	adj3
		41	adj1

Pada tabel 5.2 dan 5.3, kotak yang diberi warna biru merupakan spasi frekuensi antara kedua kota tersebut. Karena pemberian status *wanted* dan *unwanted* diset bolak-balik, maka untuk setiap dua pemancar, ada dua buah spasi frekuensi. Spasi frekeunsi final yang diambil adalah spasi frekuensi yang terbesar. Hal ini dilakukan agar spasi frekuensi tersebut dapat efektif digunakan untuk menjaga kedua pemancar tersebut bebas interferensi.

Nilai Spasi frekuensi ditentukan berdasarkan nilai *protection ratio* yang dihasilkan, kemudian mengikuti KM 15, dimana:

- Lebih dari 45 dB ➔ co channel
- Antara 33 dB s/d 45 dB ➔ adjacent-1 (minimal 100 kHz)
- Antara 7 s/d 33 dB dB ➔ adjacent-2 (minimal 200 kHz)
- Antara -7 dB s/d 7 dB ➔ adjacent-3 (minimal 300 kHz)
- Kurang dari -7 dB ➔ adjacent-4 (minimal 400 kHz)

Tabel 5.4 memperlihatkan spasi frekuensi antara masing-masing pemancar di propinsi Daerah Istimewa Yogyakarta dengan di propinsi Jawa Tengah yang juga menjadi potensi interferensi.

Tabel 5.4 Spasi Frekuensi Daerah Istimewa Yogyakarta dengan di Propinsi Jawa Tengah yang juga Menjadi Potensi Interferensi

WANTED	***UNWANTED***	SPASI FREKUENSI
KOTA YOGYAKARTA	KARANGKOBAR	1
	KEBUMAN	1
	AMBAL	2
	BRUNO	2
	KUTOARJO	2
	KLATEN	2

	MUNGKIT	2
	SUKOHARJO	2
	GROGOL	2
	GIRIWOYO	2
	TIRTOMOYO	2
	NGADIROJO	2
	JATISRUNO	1
	KARANG ANYAR	2
	KARANG PANDAN	2
	PURWODADI	*Co-channel*
	SRAGEN	1
	JUMANTORO	2
	TEMANGGUNG	2
	DEMAK	1
	SAPURAN	3
	KOTA SURAKARTA	2
	MAGELANG	2
	BANJARNEGARA	*Co-channel*
	KALI BENING	1
	SEMPOR	1
	SADANG	1
	PURWOREJO	2
	LEKSONO	1
	KALI WIRO	2
	BOYOLALI	1
	WONOGIRI	2
WONOSARI	AMBAL	1
	PURWOREJO	1
	BRUNO	1
	KUTOARJO	1
	MUNGKID	1
	BOYOLALI	1

	KLATEN	2
	SUKOHARJO	2
	GROGOL	2
	WONOGIRI	1
	EROMOKO	2
	GIRIWOYO	2
	TIRTOMOYO	2
	NGADIROJO	2
	JATISRUNO	2
	KARANG ANYAR	2
	KARANG PANDAN	2
	SRAGEN	1
	JUMANTONO	2
	TEMANGGUNG	1
	KOTA SURAKARTA	2
	KOTA MAGELANG	1
	SAPURAN	1
GAMPING	KEBUMEN	1
	KARANG KOBAR	*Co-channel*
	AMBAL	2
	SEMPOR	*Co-channel*
	SADANG	1
	PURWOREJO	2
	KUTOARJO	2
	BRUNO	2
	MUNGKID	2
	LEKSONO	1
	KALIWIRO	1
	KLATEN	2
	SUKOHARJO	2
	GROGOL	2
	NGADIROJO	1

	JUMANTONO	2
	GIRIWOYO	1
	TIRTOMOYO	1
	JATISRUNO	1
	KARANG ANYAR	1
	KARANG PANDAN	1
	SRAGEN	1
	DEMAK	*Co-channel*
	TEMANGGUNG	2
	SAPURAN	2
	KOTA MAGELANG	2
	KOTA SURAKARTA	1
BANTUL, BANGUNTAPAN	KEBUMAN	1
	PURWOREJO	1
	AMBAL	1
	KUTOARJO	2
	BOYOLALI	*Co-channel*
	BRUNO	2
	MUNGKID	2
	KLATEN	2
	SUKOHARJO	2
	GROGOL	2
	GIRIWOYO	1
	TIRTOMOYO	1
	NGADIROJO	1
	JATISRUNO	1
	KARANG ANYAR	1
	KARANG PANDAN	1
	JUMANTONO	1
	SRAGEN	1
	TEMANGGUNG	2

	MAGELANG	2
	KOTASURAKARTA	1
	SAPURAN	2
WATES, SENTOLO	KALIBENING	*Co-channel*
	KARANG KOBAR	1
	KEBUMEN	2
	SEMPOR	1
	AMBAL	2
	PURWOREJO	1
	SADANG	*Co-channel*
	KUTOARJO	2
	BRUNO	2
	LEKSONO	*Co-channel*
	KALIWIRO	*Co-channel*
	SAPURAN	2
	MUNGKIT	2
	BOYOLALI	*Co-channel*
	KLATEN	2
	SUKOHARJO	2
	GROGOL	1
	WONOGIRI	*Co-channel*
	EROMOKO	1
	GIRIWOYO	1
	TIRTOMOYO	1
	NGADIROJO	1
	JATISRUNO	*Co-channel*
	KARANG ANYAR	1
	KARANG PANDAN	1
	JUMANTORO	1
	SRAGEN	1
	TEMANGGUNG	1
	MAGELANG	2

	KOTA SURAKARTA	1
	SAPURAN	2
PATOK	AMBAL	1
	SADANG	1
	BRUNO	2
	KUTOARJO	2
	PURWOREJO	2
	KEBUMEN	1
	BOYOLALI	2
	MUNGKID	2
	LEKSONO	1
	KALIWIRO	1
	KLATEN	3
	SUKOHARJO	2
	GROGOL	2
	NGADIROJO	1
	TIRTOMOYO	2
	GIRIWOYO	2
	WONOGIRI	1
	EROMOKO	2
	JATISRUNO	2
	KARANG ANYAR	2
	KARANG PANDAN	2
	JUMANTONO	2
	SRAGEN	1
	TEMANGGUNG	2
	PURWODADI	1
	DEMAK	1
	KOTA SALATIGA	2
	KOTA SURAKARTA	2
	KOTA MAGELANG	2
	SAPURAN	1

	UNGARAN	1
	AMBARAWA	1
DEPOK	KUTOARJO	1
	AMBAL	1
	BRUNO	2
	MUNGKID	2
	KLATEN	3
	SUKOHARJO	2
	GROGOL	2
	GIRIWOYO	2
	WONOGIRI	1
	EROMOKO	1
	NGADIROJO	1
	TIRTOMOYO	2
	JATISRUNO	1
	KARANG ANYAR	2
	KARANG PANDAN	2
	JUMANTONO	2
	SRAGEN	1
	DEMAK	1
	SAPURAN	2
	TEMANGGUNG	2
	KOTA MAGELANG	2
	KOTA SURAKARTA	2
SLEMAN, NGAGLIK, KALASAN, NGEMPLAK, PAKEM	KEBUMEN	1
	BRUNO	2
	KALIWIRO	1
	LEKSONO	CO
	AMBAL	1

	MUNGKID	2
	PURWOREJO	2
	KUTOARJO	2
	KLATEN	3
	SUKOHARJO	2
	GROGOL	2
	GIRIWOYO	2
	TIRTOMOYO	2
	WONOGIRI	2
	EROMOKO	2
	NGADIROJO	1
	JATISRUNO	1
	PURWODADI	CO
	WIROSARI	1
	DEMAK	1
	KARANG ANYAR	1
	KARANG PANDAN	1
	JUMANTONO	2
	TEMANGGUNG	2
	SAPURAN	2
	KOTA MAGELANG	2
	KOTA SURAKARTA	2

Tabel 5.5 memperlihatkan spasi frekuensi antar sesama pemancar di propinsi Daerah Istimewa Yogyakarta.

Tabel 5.5 Spasi Frekuensi antar Sesama Pemancar di Propinsi Daerah Istimewa Yogyakarta

	WANTED								
UNWANTED		YOGYA	WATES	BANTUL	WONOSARI	PATOK	SLEMAN	GAMPING	DEPOK
	YOGYA		3	4	2	4	4	4	4
	WATES	3		3	2	2	2	3	2
	BANTUL	4	3		2	3	3	4	4
	WONOSARI	2	2	2		3	2	2	2
	PATOK	4	2	3	3		3	3	4
	SLEMAN	4	2	3	2	3		4	4
	GAMPING	4	3	4	2	3	4		4
	DEPOK	4	2	4	2	4	4	4	

BAB VI
ANALISIS HASIL PROGRAM OPTIMALISASI DISTRIBUSI KANAL RADIO SIARAN FM

6.1 HASIL KELUARAN PROGRAM

Keluaran dari program ini adalah semua nomor kanal yang masih dapat dialokasikan bagi masing-masing pemancar di propinsi Daerah Istimewa Yogyakarta. Nomor-nomor kanal ini dijamin bebas interferensi dengan semua nomor-nomor kanal yang sudah *exist*, yaitu yang terdaftar pada KM 15, dari semua daerah potensi interferensi yang disebutkan dalam Tabel 5.1, serta dari pemancar-pemancar dalam propinsi Daerah Istimewa Yogyakarta. Spasi frekuensi yang digunakan adalah seperti yang telah tercantum pada Tabel 5.4.

Setiap proses optimasi ini dilakukan secara terpisah, sehingga, nomor-nomor kanal yang dihasilkan merupakan hasil optimalisasi terhadap nomor-nomor kanal yang ada dalam KM15. *Output* ini akan dibandingkan dengan daftar nomor-nomor kanal untuk propinsi Daerah Istimewa Yogyakarta pada KM 15.

Tabel 6.1 memperlihatkan *output* program pada masing-masing pemancar propinsi Daerah Istimewa Yogyakarta.

Tabel 6.1 *Output* Program

Nama Pemancar	Nomor-nomor kanal
KOTA YOGYAKARTA	3, 12, 36, 52, 71 , 79, 103, 119, 138, 146, 170, 186
WATES, SENTOLO	26, 30, 60, 64, 93, 97, 127, 131, 160, 164, 194
BANTUL, BANGUNTAPAN	19, 60, 127, 194, 200
WONOSARI	14, 30, 60, 81, 97, 127, 148, 164, 194
PATOK	111

SLEMAN, NGAGLIK, KALASAN, NGEMPLAK, PAKEM	16, 32, 48, 83, 99, 115, 150, 166, 182
GAMPING	44
DEPOK	8, 27, 59, 75, 94, 126, 142, 161, 177, 193

Tabel 6.2 memperlihatkan daftar nomor-nomor kanal berdasarkan KM15.

Tabel 6.2 Daftar Nomor-nomor Kanal berdasarkan KM 15.

Nama Pemancar	*Nomor-nomor kanal*
KOTA YOGYAKARTA	4, 12, 36, 52, 71, 79, 87, 103, 119, 138, 146, 154, 170, 186
WATES, SENTOLO	63, 130, 197
BANTUL, BANGUNTAPAN	20, 67, 134, 201
WONOSARI	57, 124, 191
PATOK	111
SLEMAN, NGAGLIK, KALASAN, NGEMPLAK, PAKEM	16, 32, 48, 83, 99, 115, 150, 166, 182
GAMPING	44
DEPOK	8, 28, 75, 95, 142, 162, 178

6.2 ANALISIS HASIL KELUARAN PROGRAM

Pada Tabel 6.1 dan 6.2 diatas terlihat adanya perbedaaan yang mencolok, terutama pada pemancar Wates, Bantul, dan Wonosari. Hal ini dikarenakan spasi frekuensi yang digunakan oleh penulis dan KM 15 berbeda. Oleh karena itu, sangat penting untuk memperoleh spasi frekuensi yang tepat, yaitu spasi yang menjamin tidak hadirnya interferensi antara dua pemancar.

Perbedaan spasi frekuensi tersebut disebabkan oleh penempatan tes poin yang berbeda, sehingga kuat medan yang diterima mengalami perbedaaan dan akhirnya menghasilkan kemungkinan berbedanya hasil *protection ratio*.

6.2.1 Pemancar Kota Yogyakarta

Berdasarkan daerah potensi interferensi seperti tabel 5.1, dan spasi frekuensi pada tabel 5.4, maka dapat dianalisis bahwa dari nomor-nomor kanal yang tercantum dalam KM 15 terdapat 3 nomor yang tidak terdapat pada hasil keluaran program, yaitu 4, 87 dan 154. Artinya, nomor-nomor tersebut mengalami interferensi dengan pemancar lain. Karena semua nomot-nomor kanal yang dihasilkan oleh program juga ada pada daftar nomor kanal dari KM 15, maka tidak ada lagi nomor kanal yang dapat diberikan pada pemancar Kota Yogyakarta. Tabel 6.3 memperlihatkan hasil analisis terhadap pemancar Kota Yogyakarta.

Tabel 6.3 Analisis Nomor-nomor Kanal Pemancar Kota Yogyakarta

Nomor-nomor kanal hasil program	**3, 12, 36, 52, 71 , 79, 103, 119, 138, 146, 170, 186**
Nomor-nomor kanal berdasarkan KM15	4, 12, 36, 52, 71, 79, 87, 103, 119, 138, 146, 154, 170, 186
Nomor-nomor kanal dari KM 15 yang mengalami interferensi	4, 87, 154
Nomor-nomor kanal dari KM 15 yang bebas interferensi	12, 36, 52, 71, 79, 103, 119, 138, 146, 170, 186
Nomor-nomor kanal bebas interferensi yang masih dapat dialokasikan apabila nomor-nomor yang mengalami interferensi dihilangkan	-

6.2.2 Pemancar WATES, SENTOLO

Berdasarkan daerah potensi interferensi seperti tabel 5.1, dan spasi frekuensi pada tabel 5.4, maka dapat dianalisis bahwa dari nomor-nomor kanal yang tercantum dalam KM 15 terdapat 3 nomor yang mengalami interferensi dengan pemancar lain, yaitu 63, 130, dan 197. Karena semua nomor-nomor kanal yang dihasilkan oleh program tidak terdapat pada daftar nomor kanal

dari KM 15, maka apabila semua nomor kanal yang mengalami interferensi diabaikan semua nomor-nomor kanal dari program dapat dialokasikan untuk pemancar Wates, Sentolo, yaitu: 26, 30, 60, 64, 93, 97, 127, 131, 160, 164, dan 194. Tabel 6.4 memperlihatkan hasil analisis terhadap pemancar Wates, Sentolo.

Tabel 6.4 Analisis Nomor-nomor Kanal Pemancar WATES, SENTOLO

Nomor-nomor kanal hasil program	**26, 30, 60, 64, 93, 97, 127, 131, 160, 164, 194**
Nomor-nomor kanal berdasarkan KM15	**63, 130, 197**
Nomor-nomor kanal dari KM 15 yang mengalami interferensi	**63, 130, 197**
Nomor-nomor kanal dari KM 15 yang bebas interferensi	**-**
Nomor-nomor kanal bebas interferensi yang masih dapat dialokasikan apabila nomor-nomor yang mengalami interferensi dihilangkan	**26, 30, 60, 64, 93, 97, 127, 131, 160, 164, 194**

6.2.3 Pemancar BANTUL, BANGUNTAPAN

Berdasarkan daerah potensi interferensi seperti tabel 5.1, dan spasi frekuensi pada tabel 5.4, maka dapat dianalisis bahwa dari nomor-nomor kanal yang tercantum dalam KM 15 terdapat 4 nomor yang mengalami interferensi dengan pemancar lain, yaitu 20, 67, 134, dan 201. Karena semua nomor-nomor kanal yang dihasilkan oleh program tidak terdapat pada daftar nomor kanal dari KM 15, maka apabila semua nomor kanal yang mengalami interferensi diabaikan semua nomor-nomor kanal dari program dapat dialokasikan untuk pemancar Bantul, Banguntapan, yaitu: 19, 60, 127, 194, dan 200. Tabel 6.5 memperlihatkan hasil analisis terhadap pemancar Bantul, Banguntapan.

Tabel 6.5 Analisis Nomor-nomor Kanal Pemancar BANTUL, BANGUNTAPAN

Nomor-nomor kanal hasil program	**19, 60, 127, 194, 200**
Nomor-nomor kanal berdasarkan KM15	**20, 67, 134, 201**
Nomor-nomor kanal dari KM 15 yang mengalami interferensi	**20, 67, 134, 201**
Nomor-nomor kanal dari KM 15 yang bebas interferensi	**-**
Nomor-nomor kanal bebas interferensi yang masih dapat dialokasikan apabila nomor-nomor yang mengalami interferensi dihilangkan	**19, 60, 127, 194, 200**

6.2.4 Pemancar WONOSARI

Berdasarkan daerah potensi interferensi seperti tabel 5.1, dan spasi frekuensi pada tabel 5.4, maka dapat dianalisis bahwa dari nomor-nomor kanal yang tercantum dalam KM 15 untuk pemancar Wonosari, terdapat 3 nomor yang mengalami interferensi dengan pemancar lain, yaitu 57, 124, dan 191. Karena semua nomor-nomor kanal yang dihasilkan oleh program tidak terdapat pada daftar nomor kanal dari KM 15, maka apabila semua nomor kanal yang mengalami interferensi diabaikan semua nomor-nomor kanal dari program dapat dialokasikan untuk pemancar Wonosari, yaitu: 14, 30, 60, 81, 97, 127, 148, 164, dan 194. Tabel 6.6 memperlihatkan hasil analisis terhadap pemancar Wonosari.

Tabel 6.6 Analisis Nomor-nomor Kanal Pemancar WONOSARI

Nomor-nomor kanal hasil program	14, 30, 60, 81, 97, 127, 148, 164, 194
Nomor-nomor kanal berdasarkan KM15	57, 124, 191
Nomor-nomor kanal dari KM 15 yang mengalami interferensi	57, 124, 191

Nomor-nomor kanal dari KM 15 yang bebas interferensi	-
Nomor-nomor kanal bebas interferensi yang masih dapat dialokasikan apabila nomor-nomor yang mengalami interferensi dihilangkan	14, 30, 60, 81, 97, 127, 148, 164, 194

6.2.5 Pemancar PATOK

Berdasarkan daerah potensi interferensi seperti tabel 5.1, dan spasi frekuensi pada tabel 5.4, maka dapat dianalisis bahwa tidak ada nomor kanal dari daftar KM 15 untuk pemancar Patok yang mengalami interferensi dengan pemancar lain. Keluaran dari program memperlihatkan nomor kanal yang sama dengan KM 15, yaitu 111. Dengan demikian tidak ada lagi kanal yang bisa diberikan untuk pemancar Patok. Hasil analisis terhadap pemancar Patok dapat dilihat pada Tabel 6.7.

Tabel 6.7 Analisis Nomor-nomor Kanal Pemancar PATOK

Nomor-nomor kanal hasil program	111
Nomor-nomor kanal berdasarkan KM15	111
Nomor-nomor kanal dari KM 15 yang mengalami interferensi	-
Nomor-nomor kanal dari KM 15 yang bebas interferensi	111
Nomor-nomor kanal bebas interferensi yang masih dapat dialokasikan apabila nomor-nomor yang mengalami interferensi dihilangkan	-

6.2.6 Pemancar SLEMAN, NGAGLIK, KALASAN, NGEMPLAK, PAKEM

Berdasarkan daerah potensi interferensi seperti tabel 5.1, dan spasi frekuensi pada tabel 5.4, maka dapat dianalisis bahwa tidak ada nomor kanal dari daftar KM 15 untuk pemancar Sleman, Ngaglik, Kalasan, Ngemplak, Pakem yang mengalami interferensi dengan pemancar lain. Keluaran dari program memperlihatkan nomor kanal yang sama dengan KM 15, yaitu 16,

32, 48, 83, 99, 115, 150, 166, dan 182. Dengan demikian tidak ada lagi kanal yang bisa diberikan untuk pemancar Sleman, Ngaglik, Kalasan, Ngemplak, Pakem. Hasil analisis ini dapat dilihat pada Tabel 6.8.

Tabel 6.8 Analisis Nomor-nomor Kanal Pemancar SLEMAN, NGAGLIK, KALASAN, NGEMPLAK, PAKEM

Nomor-nomor kanal hasil program	**16, 32, 48, 83, 99, 115, 150, 166, 182**
Nomor-nomor kanal berdasarkan KM15	**16, 32, 48, 83, 99, 115, 150, 166, 182**
Nomor-nomor kanal dari KM 15 yang mengalami interferensi	**-**
Nomor-nomor kanal dari KM 15 yang bebas interferensi	**16, 32, 48, 83, 99, 115, 150, 166, 182**
Nomor-nomor kanal bebas interferensi yang masih dapat dialokasikan apabila nomor-nomor yang mengalami interferensi dihilangkan	**-**

6.2.7 Pemancar GAMPING

Berdasarkan daerah potensi interferensi seperti tabel 5.1, dan spasi frekuensi pada tabel 5.4, maka dapat dianalisis bahwa tidak ada nomor kanal dari daftar KM 15 untuk pemancar Gamping yang mengalami interferensi dengan pemancar lain. Keluaran dari program memperlihatkan nomor kanal yang sama dengan KM 15, yaitu nomor 44. Dengan demikian tidak ada lagi kanal yang bisa diberikan untuk pemancar Gamping. Hasil analisis ini dapat dilihat pada Tabel 6.9.

Tabel 6.9 Analisis Nomor-nomor Kanal Pemancar GAMPING

Nomor-nomor kanal hasil program	**44**
Nomor-nomor kanal berdasarkan KM15	44

Nomor-nomor kanal dari KM 15 yang mengalami interferensi	-
Nomor-nomor kanal dari KM 15 yang bebas interferensi	44
Nomor-nomor kanal bebas interferensi yang masih dapat dialokasikan apabila nomor-nomor yang mengalami interferensi dihilangkan	-

6.2.8 Pemancar DEPOK

Berdasarkan daerah potensi interferensi seperti tabel 5.1, dan spasi frekuensi pada tabel 5.4, maka dapat dianalisis bahwa dari nomor-nomor kanal yang ada pada daftar KM 15, 4 kanal mengalami interferensi dengan pemancar lain, yaitu: 28, 95, 162, 178. Apabila nomor-nomor kanal yang mengalami interferensi dihilangkan dari daftar, maka terdapat 7 nomor kanal dari keluaran program yang dapat dialokasikan untuk pemancar Depok. Nomor-nomor kanal tersebut adalah 27, 59, 94, 126, 161, 177, dan 193. Hasil analisis dapat dilihat pada Tabel 6.10.

Tabel 6.10 Analisis Nomor-nomor Kanal Pemancar DEPOK

Nomor-nomor kanal hasil program	**8, 27, 59, 75, 94, 126, 142, 161, 177, 193**
Nomor-nomor kanal berdasarkan KM15	8, 28, 75, 95, 142, 162, 178
Nomor-nomor kanal dari KM 15 yang mengalami interferensi	28, 95, 162, 178
Nomor-nomor kanal dari KM 15 yang bebas interferensi	8, 75, 142
Nomor-nomor kanal bebas interferensi yang masih dapat dialokasikan apabila nomor-nomor yang mengalami interferensi dihilangkan	27, 59, 94, 126, 161, 177, 193

6.2.9 Rangkuman Analisa Pemancar-pemancar pada Propinsi D.I Yogyakarta

Tabel 6.11 memperlihatkan daftar pemancar beserta nomor-nomor kanal yang mengalami interferensi dari seluruh pemancar di propinsi Daerah Istimewa Yogyakarta.

Tabel 6.11 Pemancar-pemancar yang Mengalami Interferensi

Nama Pemancar	Nomor-nomor kanal
Kota Yogyakarta	**4, 87, 154,**
WATES, SENTOLO	**63, 130, 197**
BANTUL, BANGUNTAPAN	**20, 67, 134, 201**
WONOSARI	**57, 124, 191**
DEPOK	**28, 95, 162, 178**

Tabel 6.12 memperlihatkan daftar nomor kanal yang dapat dialokasikan selain yang sudah *exist* dari seluruh pemancar di propinsi Daerah Istimewa Yogyakarta, apabila nomor kanal yang mengalami interferensi dihilangkan.

Tabel 6.12 Pemancar-pemancar yang Masih Mungkin Mendapatkan Tambahan Kanal

Nama Pemancar	Nomor-nomor kanal
Kota Yogyakarta	**3**
WATES, SENTOLO	26, 30, 60, 64, 93, 97, 127, 131, 160, 164, 194
BANTUL, BANGUNTAPAN	19, 60, 127, 194, 200
WONOSARI	14, 30, 60, 81, 97, 127, 148, 164, 194
DEPOK	27, 59, 94, 126, 161, 177, 193

BAB VII
KESIMPULAN

Dari hasil analisis yang dilakukan, maka terdapat nomor-nomor kanal yang mengalami interferensi dalam propinsi Daerah Istimewa Yogyakarta berdasarkan KM 15. Pemancar-pemancar tersebut adalah pemancar Kota Yogyakarta; WATES, SENTOLO; BANTUL, BANGUNTAPAN; WONOSARI; DEPOK.

Pemancar-pemancar yang kepadanya masih dapat dialokasikan kanal-kanal baru adalah sebagai berikut (beserta nomor-nomor kanal yang mungkin untuk dialokasikan): Kota Yogyakarta, yaitu 3; WATES, SENTOLO, yaitu 26, 30, 60, 64, 93, 97, 127, 131, 160, 164, 194; BANTUL, BANGUNTAPAN, yaitu 19, 60, 127, 194, 200; WONOSARI, yaitu 14, 30, 60, 81, 97, 127, 148, 164, 194; DEPOK, yaitu 27, 59, 94, 126, 161, 177, 193.

Dapat disimpulkan bahwa distribusi kanal yang dikerjakan oleh KM 15 untuk wilayah propinsi Daerah Istimewa Yogyakarta selama ini kurang optimal, karena masih memiliki beberapa nomor kanal yang mengalami interferensi.

DAFTAR ACUAN

[1] ”____”, ”***Prinsip perencanaan frekuensi Radio Siaran FM di Indonesia***”, Rapat Kerja Teknis/ Lokakarya KPI – KPID, Bandung, 2 Desember 2004, Ditjen Postel.

[2] ”____”,”***MASTER PLAN PENETAPAN FREKUENSI KANAL RADIO SIARAN FM***”, Ditjen Postel – DEPKOMINFO, Jakarta, Juli 2005

[3] ”____”, ***Planning Standards for Terrestrial FM Sound Broadcasting at VHF***”, Rekomendasi ITU-R BS.412-9*, tahun 1998.

[4] “____”, “ ***Determination af the interference field strength in the Land Mobile Service***”, Rekomendasi ITU-R P.370, Annex 5, tahun 1998.

[5] “____”,“***METHOD OF MEASURING THE MAXIMUM FREQUENCY DEVIATION OF FM BROADCAST EMISSIONS IN THE BAND 87.5 MHz TO 108 MHz AT MONITORING STATIONS***”, CEPT/ERC Recommendation ERC 54-01 E (Funchal 1998), tahun 1998.

[6] KEPUTUSAN MENTRI NO.15 Tahun 2003 tentang RENCANA INDUK FREKUENSI RADIO PENYELENGGARAAN TELEKOMUNIKASI KHUSU UNTUK KEPERLUAN RADIO SIARAN FM.

[7] “____”,”**BC-Training**”, LS Telcom Databases, tahun 2006.

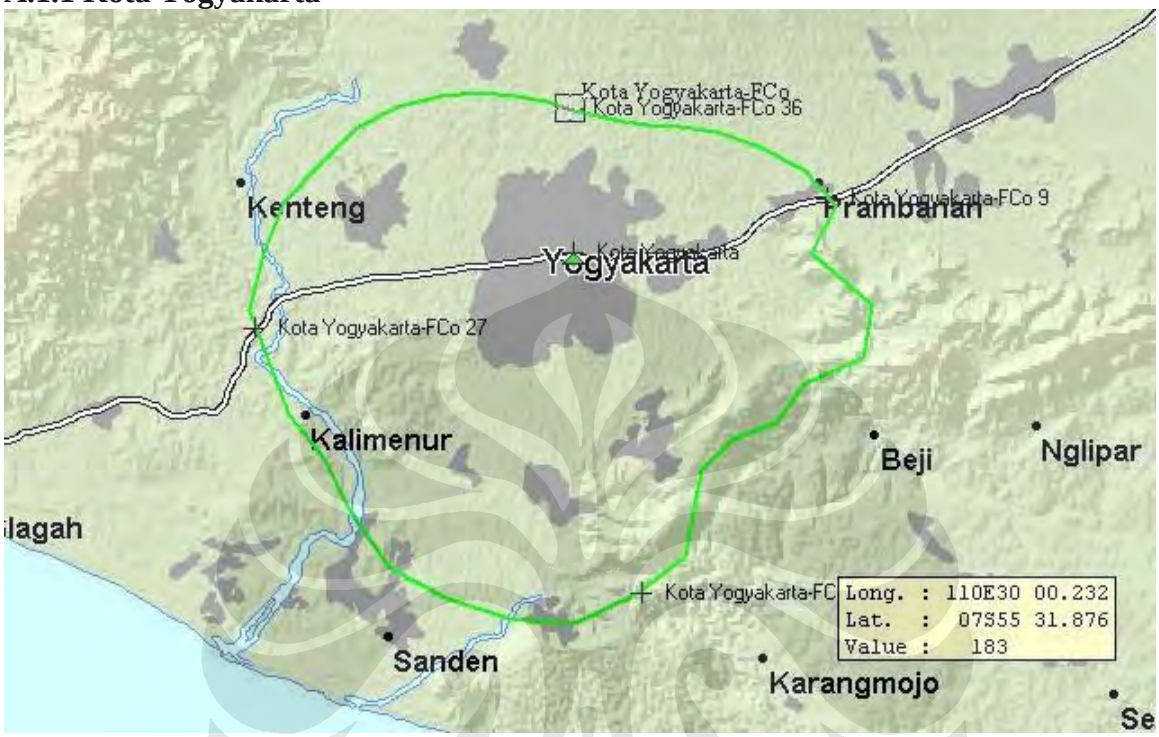
DAFTAR PUSTAKA

Deitel, “JAVA How To Program Fifth Edition”, Prantice Hall, 2003.

LAMPIRAN A
SIMULASI CONTOUR CALCULATION

A.1 SIMULASI KONTUR DAN TES POIN

A.1.1 Kota Yogyakarta

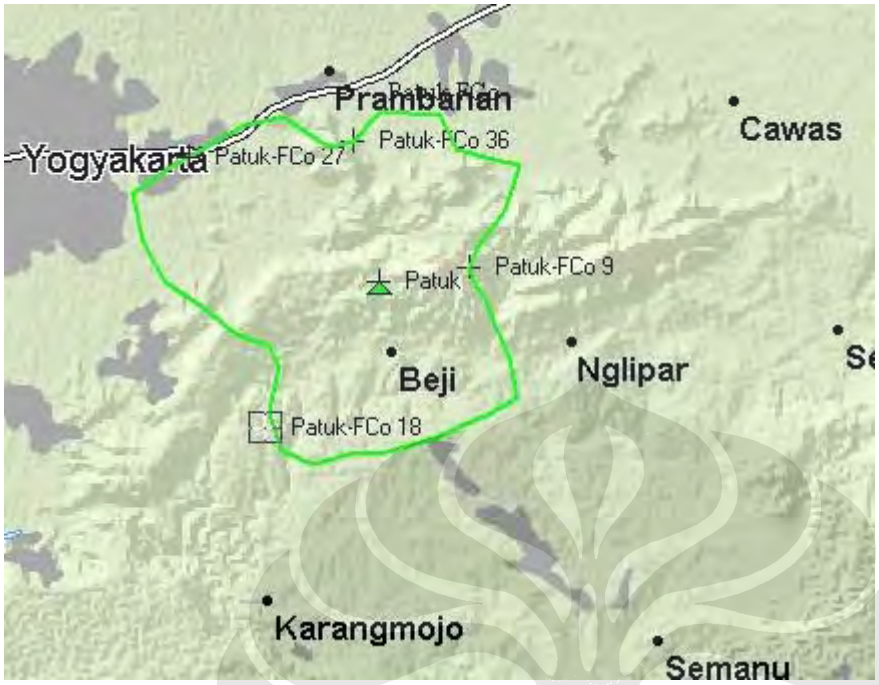


A.1.2 Wates, Sentolo

A.1.3 BANTUL, BANGUNTAPAN

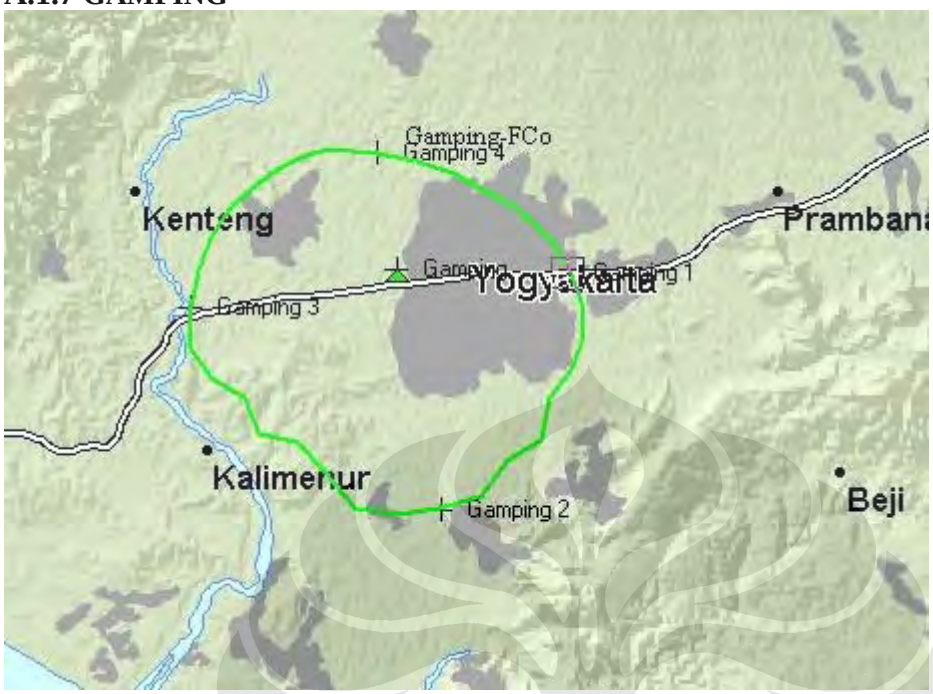
A.1.4 WONOSARI

A.1.5 PATOK

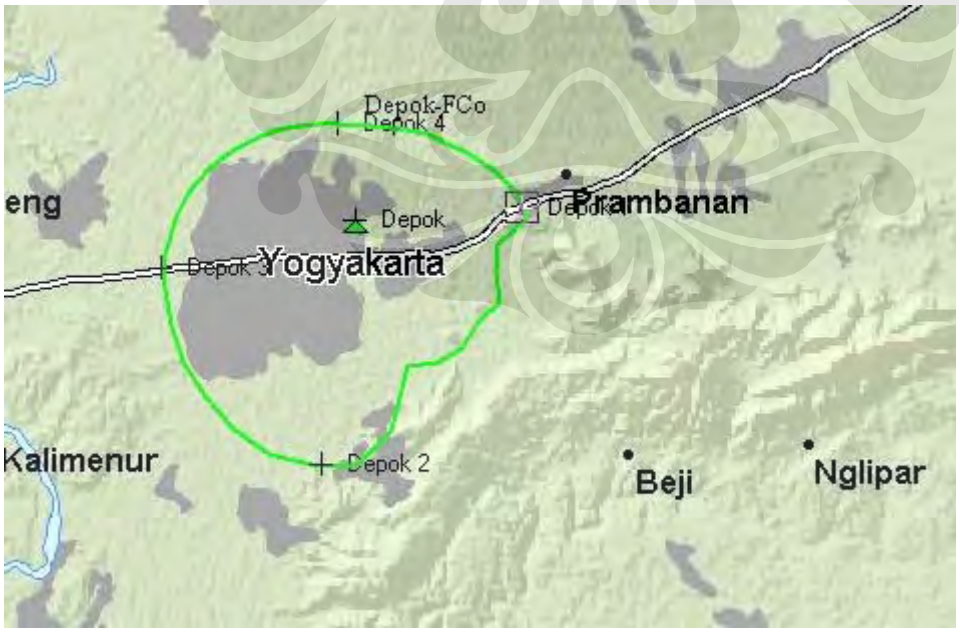


A.1.6 SLEMAN, NGAGLIK, KALASAN, NGEMPLAK, PAKEM

A.1.7 GAMPING



A.1.8 DEPOK



A.2 PERHITUNGAN KONTUR

A.2.1 Kota Yogyakarta

User : sysadmin
Calculation Mode : Fieldstrength Contour GE 84
Calculation Model : ITU-R 1546 Database E50 Stereo
Receiver height : 10.0
Radio Service : FM Broadcast
Clear. Angle Corr.: Yes
Use Land-Sea Disc.: Yes
Calculated : 16/04/2008 23:12 CHIRplus_BC V 4.4.2

Wanted Transmit. : Kota Yogyakarta
Frequency/MHz : 100.000 Chan. :
MaxERP kW / dBkW : 2.700 / 4.314
Longit. / Latit. : 110E23 12 / 07S46 59
Heff Max : 147 Country: INS
Polarisation : V OS :
Antenna Dir : ND Service: FM Broadcast
Offset : 0 System : 4

Fieldstrength Contour for Transmitter : Kota Yogyakarta

AZM	LONGITUDE	LATITUDE	DIST	VALUE	AZM	LONGITUDE	LATITUDE	DIST	VALUE
0.0	110E23 12	07S42 45	7.8	66.0	180.0	110E23 12	07S58 02	20.5	66.0
10.0	110E23 56	07S42 52	7.7	66.0	190.0	110E21 16	07S57 52	20.5	66.0
20.0	110E24 39	07S43 02	7.8	66.0	200.0	110E19 25	07S57 16	20.3	66.0
30.0	110E25 26	07S43 08	8.2	66.0	210.0	110E17 44	07S56 22	20.1	66.0
40.0	110E26 24	07S43 12	9.1	66.0	220.0	110E16 36	07S54 47	18.9	66.0
50.0	110E27 31	07S43 23	10.4	66.0	230.0	110E15 45	07S53 10	17.9	66.0
60.0	110E28 46	07S43 48	11.8	66.0	240.0	110E14 42	07S51 50	18.0	66.0
70.0	110E30 05	07S44 30	13.4	66.0	250.0	110E14 06	07S50 16	17.8	66.0
80.0	110E31 05	07S45 36	14.7	66.0	260.0	110E13 30	07S48 40	18.1	66.0
90.0	110E30 20	07S46 59	13.1	66.0	270.0	110E13 57	07S46 59	17.0	66.0
100.0	110E32 08	07S48 32	16.7	66.0	280.0	110E14 33	07S45 28	16.1	66.0
110.0	110E31 54	07S50 07	17.0	66.0	290.0	110E15 41	07S44 16	14.7	66.0
120.0	110E30 06	07S50 55	14.6	66.0	300.0	110E16 41	07S43 15	13.8	66.0
130.0	110E29 16	07S52 01	14.5	66.0	310.0	110E17 57	07S42 36	12.6	66.0
140.0	110E27 56	07S52 34	13.5	66.0	320.0	110E19 14	07S42 17	11.3	66.0
150.0	110E27 00	07S53 30	14.0	66.0	330.0	110E20 26	07S42 13	10.2	66.0
160.0	110E26 33	07S56 05	18.0	66.0	340.0	110E21 29	07S42 18	9.2	66.0
170.0	110E25 01	07S57 12	19.2	66.0	350.0	110E22 24	07S42 29	8.5	66.0

Covered Area : 692.75 sqkm

A.W.2 Wates, Sentolo

User : sysadmin
Calculation Mode : Fieldstrength Contour GE 84
Calculation Model : ITU-R 1546 Database E50 Stereo
Receiver height : 10.0
Radio Service : FM Broadcast
Clear. Angle Corr.: Yes
Use Land-Sea Disc.: Yes
Calculated : 16/04/2008 23:12 CHIRplus_BC V 4.4.2

Wanted Transmit. : Wates,Sentolo
Frequency/MHz : 100.000 Chan. :
MaxERP kW / dBkW : 2.000 / 3.010
Longit. / Latit. : 110E11 44 / 07S52 10
Heff Max : 62 Country: INS
Polarisation : V OS :
Antenna Dir : ND Service: FM Broadcast
Offset : 0 System : 4

Fieldstrength Contour for Transmitter : Wates,Sentolo

AZM	LONGITUDE	LATITUDE	DIST	VALUE	AZM	LONGITUDE	LATITUDE	DIST	VALUE
0.0	110E11 44	07S48 33	6.7	66.0	180.0	110E11 44	07S58 27	11.7	66.0
10.0	110E12 26	07S48 14	7.4	66.0	190.0	110E10 36	07S58 33	12.0	66.0
20.0	110E13 06	07S48 27	7.3	66.0	200.0	110E09 30	07S58 15	12.0	66.0
30.0	110E13 47	07S48 39	7.5	66.0	210.0	110E08 29	07S57 44	11.9	66.0
40.0	110E14 24	07S49 00	7.6	66.0	220.0	110E07 39	07S56 59	11.7	66.0
50.0	110E14 56	07S49 30	7.7	66.0	230.0	110E06 54	07S56 11	11.6	66.0
60.0	110E15 12	07S50 11	7.4	66.0	240.0	110E06 28	07S55 11	11.2	66.0
70.0	110E15 24	07S50 50	7.2	66.0	250.0	110E06 01	07S54 13	11.2	66.0
80.0	110E15 46	07S51 28	7.5	66.0	260.0	110E05 49	07S53 12	11.0	66.0
90.0	110E15 59	07S52 10	7.8	66.0	270.0	110E06 21	07S52 10	9.9	66.0
100.0	110E16 20	07S52 58	8.6	66.0	280.0	110E07 43	07S51 28	7.5	66.0
110.0	110E16 41	07S53 57	9.7	66.0	290.0	110E08 34	07S51 01	6.2	66.0
120.0	110E16 28	07S54 52	10.0	66.0	300.0	110E09 07	07S50 40	5.5	66.0
130.0	110E15 37	07S55 24	9.3	66.0	310.0	110E09 31	07S50 19	5.3	66.0
140.0	110E15 13	07S56 17	10.0	66.0	320.0	110E10 07	07S50 15	4.6	66.0
150.0	110E14 31	07S56 56	10.2	66.0	330.0	110E10 18	07S49 43	5.2	66.0
160.0	110E13 45	07S57 40	10.8	66.0	340.0	110E10 48	07S49 37	5.0	66.0
170.0	110E12 48	07S58 13	11.4	66.0	350.0	110E11 12	07S49 10	5.6	66.0

Covered Area : 252.72 sqkm

A.2.3 BANTUL, BANGUNTAPAN

User : sysadmin
Calculation Mode : Fieldstrength Contour GE 84
Calculation Model : ITU-R 1546 Database E50 Stereo
Receiver height : 10.0
Radio Service : FM Broadcast
Clear. Angle Corr.: Yes
Use Land-Sea Disc.: Yes
Calculated : 16/04/2008 23:12 CHIRplus_BC V 4.4.2

Wanted Transmit. : Bantul,Banguntapan
Frequency/MHz : 100.000 Chan. :
MaxERP kW / dBkW : 2.000 / 3.010
Longit. / Latit. : 110E22 07 / 07S52 01
Heff Max : 67 Country: INS
Polarisation : V OS :
Antenna Dir : ND Service: FM Broadcast
Offset : 0 System : 4

Fieldstrength Contour for Transmitter : Bantul,Banguntapan

AZM	LONGITUDE	LATITUDE	DIST	VALUE	AZM	LONGITUDE	LATITUDE	DIST	VALUE
0.0	110E22 07	07S48 13	7.0	66.0	180.0	110E22 07	07S55 36	6.6	66.0
10.0	110E22 46	07S48 19	7.0	66.0	190.0	110E21 10	07S57 21	10.0	66.0
20.0	110E23 26	07S48 26	7.1	66.0	200.0	110E19 48	07S58 20	12.4	66.0
30.0	110E24 07	07S48 35	7.4	66.0	210.0	110E18 43	07S57 50	12.4	66.0
40.0	110E24 48	07S48 51	7.7	66.0	220.0	110E17 51	07S57 03	12.2	66.0
50.0	110E25 31	07S49 12	8.1	66.0	230.0	110E17 23	07S55 57	11.3	66.0
60.0	110E26 06	07S49 45	8.5	66.0	240.0	110E16 35	07S55 11	11.7	66.0
70.0	110E26 40	07S50 23	8.9	66.0	250.0	110E16 26	07S54 04	11.1	66.0
80.0	110E25 39	07S51 24	6.6	66.0	260.0	110E16 57	07S52 56	9.6	66.0
90.0	110E25 15	07S52 01	5.8	66.0	270.0	110E17 08	07S52 01	9.1	66.0
100.0	110E25 30	07S52 37	6.3	66.0	280.0	110E17 37	07S51 14	8.4	66.0
110.0	110E25 08	07S53 07	5.9	66.0	290.0	110E18 02	07S50 33	8.0	66.0
120.0	110E24 49	07S53 34	5.7	66.0	300.0	110E18 08	07S49 45	8.5	66.0
130.0	110E24 27	07S53 58	5.6	66.0	310.0	110E18 39	07S49 09	8.3	66.0
140.0	110E24 10	07S54 27	5.9	66.0	320.0	110E19 18	07S48 42	8.0	66.0
150.0	110E23 53	07S55 03	6.5	66.0	330.0	110E20 02	07S48 27	7.6	66.0
160.0	110E23 21	07S55 22	6.6	66.0	340.0	110E20 45	07S48 18	7.4	66.0
170.0	110E22 48	07S55 52	7.2	66.0	350.0	110E21 26	07S48 12	7.2	66.0

Covered Area : 218.74 sqkm

A.2.4 WONOSARI

User : sysadmin
Calculation Mode : Fieldstrength Contour GE 84
Calculation Model : ITU-R 1546 Database E50 Stereo
Receiver height : 10.0
Radio Service : FM Broadcast
Clear. Angle Corr.: Yes
Use Land-Sea Disc.: Yes
Calculated : 16/04/2008 23:12 CHIRplus_BC V 4.4.2

Wanted Transmit. : Wonosari
Frequency/MHz : 100.000 Chan. :
MaxERP kW / dBkW : 2.000 / 3.010
Longit. / Latit. : 110E35 33 / 07S58 12
Heff Max : 64 Country: INS
Polarisation : V OS :
Antenna Dir : ND Service: FM Broadcast
Offset : 0 System : 4

Fieldstrength Contour for Transmitter : Wonosari

AZM	LONGITUDE	LATITUDE	DIST	VALUE	AZM	LONGITUDE	LATITUDE	DIST	VALUE
0.0	110E35 33	07S53 48	8.1	66.0	180.0	110E35 33	08S02 52	8.7	66.0
10.0	110E36 13	07S54 31	6.9	66.0	190.0	110E34 41	08S03 03	9.1	66.0
20.0	110E37 03	07S54 08	8.0	66.0	200.0	110E33 52	08S02 46	9.0	66.0
30.0	110E38 10	07S53 42	9.6	66.0	210.0	110E33 16	08S02 06	8.4	66.0
40.0	110E38 57	07S54 12	9.7	66.0	220.0	110E33 01	08S01 12	7.3	66.0
50.0	110E39 20	07S55 03	9.1	66.0	230.0	110E32 31	08S00 43	7.3	66.0
60.0	110E39 41	07S55 50	8.7	66.0	240.0	110E31 48	08S00 21	8.0	66.0
70.0	110E39 25	07S56 48	7.5	66.0	250.0	110E30 53	07S59 53	9.1	66.0
80.0	110E40 10	07S57 23	8.6	66.0	260.0	110E29 10	07S59 19	11.9	66.0
90.0	110E40 09	07S58 12	8.5	66.0	270.0	110E28 55	07S58 12	12.2	66.0
100.0	110E40 07	07S59 00	8.5	66.0	280.0	110E29 26	07S57 08	11.4	66.0
110.0	110E39 39	07S59 41	8.0	66.0	290.0	110E30 20	07S56 19	10.2	66.0
120.0	110E39 42	08S00 34	8.8	66.0	300.0	110E31 06	07S55 39	9.4	66.0
130.0	110E39 00	08S01 04	8.3	66.0	310.0	110E31 22	07S54 43	10.0	66.0
140.0	110E38 20	08S01 28	7.9	66.0	320.0	110E32 11	07S54 13	9.6	66.0
150.0	110E37 45	08S01 59	8.1	66.0	330.0	110E33 02	07S53 52	9.3	66.0
160.0	110E37 02	08S02 14	8.0	66.0	340.0	110E34 05	07S54 12	7.9	66.0
170.0	110E36 19	08S02 27	8.0	66.0	350.0	110E34 54	07S54 33	6.9	66.0

Covered Area : 244.08 sqkm

A.2.5 PATOK

User : sysadmin
Calculation Mode : Fieldstrength Contour GE 84
Calculation Model : ITU-R 1546 Database E50 Stereo
Receiver height : 10.0
Radio Service : FM Broadcast
Clear. Angle Corr.: Yes
Use Land-Sea Disc.: Yes
Calculated : 16/04/2008 23:12 CHIRplus_BC V 4.4.2

Wanted Transmit. : Patuk
Frequency/MHz : 100.000 Chan. :
MaxERP kW / dBkW : 0.140 / -8.539
Longit. / Latit. : 110E31 52 / 07S50 33
Heff Max : 298 Country: INS
Polarisation : V OS :
Antenna Dir : ND Service: FM Broadcast
Offset : 0 System : 4

Fieldstrength Contour for Transmitter : Patuk

AZM	LONGITUDE	LATITUDE	DIST	VALUE	AZM	LONGITUDE	LATITUDE	DIST	VALUE
0.0	110E31 52	07S46 02	8.4	66.0	180.0	110E31 52	07S55 08	8.5	66.0
10.0	110E32 41	07S46 01	8.5	66.0	190.0	110E31 03	07S55 10	8.7	66.0
20.0	110E33 31	07S46 03	8.9	66.0	200.0	110E30 05	07S55 25	9.6	66.0
30.0	110E33 58	07S46 57	7.7	66.0	210.0	110E29 15	07S55 02	9.6	66.0
40.0	110E34 43	07S47 11	8.1	66.0	220.0	110E28 56	07S54 01	8.4	66.0
50.0	110E35 39	07S47 25	9.1	66.0	230.0	110E29 10	07S52 48	6.5	66.0
60.0	110E35 16	07S48 36	7.2	66.0	240.0	110E28 56	07S52 14	6.2	66.0
70.0	110E34 32	07S49 35	5.2	66.0	250.0	110E28 00	07S51 57	7.6	66.0
80.0	110E34 17	07S50 08	4.5	66.0	260.0	110E27 18	07S51 21	8.5	66.0
90.0	110E34 23	07S50 33	4.6	66.0	270.0	110E26 06	07S50 33	10.6	66.0
100.0	110E34 49	07S51 04	5.5	66.0	280.0	110E25 31	07S49 26	11.8	66.0
110.0	110E35 01	07S51 41	6.1	66.0	290.0	110E25 16	07S48 10	12.9	66.0
120.0	110E35 23	07S52 34	7.4	66.0	300.0	110E26 21	07S47 24	11.7	66.0
130.0	110E35 37	07S53 40	9.0	66.0	310.0	110E27 24	07S46 50	10.7	66.0
140.0	110E34 52	07S54 05	8.6	66.0	320.0	110E28 17	07S46 19	10.2	66.0
150.0	110E34 06	07S54 23	8.2	66.0	330.0	110E29 18	07S46 08	9.5	66.0
160.0	110E33 24	07S54 44	8.2	66.0	340.0	110E30 32	07S46 55	7.2	66.0
170.0	110E32 39	07S54 55	8.2	66.0	350.0	110E31 12	07S46 46	7.1	66.0

Covered Area : 224.73 sqkm

A.2.6 SLEMAN, NGAGLIK, KALASAN, NGEMPLAK, PAKEM

User : sysadmin
Calculation Mode : Fieldstrength Contour GE 84
Calculation Model : ITU-R 1546 Database E50 Stereo
Receiver height : 10.0
Radio Service : FM Broadcast
Clear. Angle Corr.: Yes
Use Land-Sea Disc.: Yes
Calculated : 16/04/2008 23:12 CHIRplus_BC V 4.4.2

Wanted Transmit. : Sleman,Ngaglik,Kalasan,Ngemplak
Frequency/MHz : 100.000 Chan. :
MaxERP kW / dBkW : 0.275 / -5.607
Longit. / Latit. : 110E25 16 / 07S41 51
Heff Max : 205 Country: INS
Polarisation : V OS :
Antenna Dir : ND Service: FM Broadcast
Offset : 0 System : 4

Fieldstrength Contour for Transmitter : Sleman,Ngaglik,Kalasan,Ngemplak

AZM	LONGITUDE	LATITUDE	DIST	VALUE	AZM	LONGITUDE	LATITUDE	DIST	VALUE
0.0	110E25 16	07S39 50	3.7	66.0	180.0	110E25 16	07S48 34	12.5	66.0
10.0	110E25 37	07S39 52	3.7	66.0	190.0	110E24 06	07S48 21	12.2	66.0
20.0	110E25 58	07S39 56	3.8	66.0	200.0	110E23 02	07S47 57	12.0	66.0
30.0	110E26 20	07S40 01	3.9	66.0	210.0	110E22 09	07S47 12	11.5	66.0
40.0	110E26 43	07S40 08	4.1	66.0	220.0	110E21 23	07S46 25	11.1	66.0
50.0	110E27 06	07S40 20	4.4	66.0	230.0	110E20 48	07S45 34	10.7	66.0
60.0	110E27 31	07S40 34	4.8	66.0	240.0	110E20 32	07S44 33	10.0	66.0
70.0	110E28 00	07S40 52	5.4	66.0	250.0	110E20 29	07S43 34	9.3	66.0
80.0	110E28 34	07S41 16	6.2	66.0	260.0	110E20 46	07S42 38	8.4	66.0
90.0	110E29 06	07S41 51	7.0	66.0	270.0	110E21 07	07S41 51	7.6	66.0
100.0	110E29 33	07S42 36	8.0	66.0	280.0	110E21 38	07S41 13	6.8	66.0
110.0	110E29 55	07S43 31	9.1	66.0	290.0	110E22 10	07S40 44	6.1	66.0
120.0	110E29 57	07S44 31	9.9	66.0	300.0	110E22 44	07S40 24	5.4	66.0
130.0	110E29 20	07S45 14	9.8	66.0	310.0	110E23 14	07S40 09	4.9	66.0
140.0	110E28 53	07S46 07	10.3	66.0	320.0	110E23 40	07S39 58	4.6	66.0
150.0	110E28 26	07S47 18	11.7	66.0	330.0	110E24 06	07S39 51	4.3	66.0
160.0	110E27 32	07S48 01	12.2	66.0	340.0	110E24 31	07S39 48	4.0	66.0
170.0	110E26 27	07S48 28	12.5	66.0	350.0	110E24 54	07S39 47	3.9	66.0

Covered Area : 212.89 sqkm

A.2.7 GAMPING

User : sysadmin
Calculation Mode : Fieldstrength Contour GE 84
Calculation Model : ITU-R 1546 Database E50 Stereo
Receiver height : 10.0
Radio Service : FM Broadcast
Clear. Angle Corr.: Yes
Use Land-Sea Disc.: Yes
Calculated : 16/04/2008 23:12 CHIRplus_BC V 4.4.2

Wanted Transmit. : Gamping
Frequency/MHz : 100.000 Chan. :
MaxERP kW / dBkW : 1.100 / 0.414
Longit. / Latit. : 110E20 19 / 07S47 00
Heff Max : 92 Country: INS
Polarisation : V OS :
Antenna Dir : ND Service: FM Broadcast
Offset : 0 System : 4

Fieldstrength Contour for Transmitter : Gamping

AZM	LONGITUDE	LATITUDE	DIST	VALUE	AZM	LONGITUDE	LATITUDE	DIST	VALUE
0.0	110E20 19	07S43 59	5.6	66.0	180.0	110E20 19	07S53 32	12.1	66.0
10.0	110E20 50	07S44 03	5.6	66.0	190.0	110E19 11	07S53 21	11.9	66.0
20.0	110E21 20	07S44 14	5.4	66.0	200.0	110E18 20	07S52 24	10.6	66.0
30.0	110E21 49	07S44 25	5.5	66.0	210.0	110E17 37	07S51 37	9.9	66.0
40.0	110E22 18	07S44 39	5.7	66.0	220.0	110E16 35	07S51 24	10.6	66.0
50.0	110E22 48	07S44 56	6.0	66.0	230.0	110E16 13	07S50 25	9.8	66.0
60.0	110E23 21	07S45 16	6.4	66.0	240.0	110E15 18	07S49 52	10.6	66.0
70.0	110E23 54	07S45 42	7.0	66.0	250.0	110E14 42	07S49 02	11.0	66.0
80.0	110E24 29	07S46 17	7.8	66.0	260.0	110E14 45	07S47 59	10.4	66.0
90.0	110E24 54	07S47 00	8.4	66.0	270.0	110E14 58	07S47 00	9.8	66.0
100.0	110E25 14	07S47 52	9.2	66.0	280.0	110E15 18	07S46 08	9.3	66.0
110.0	110E25 19	07S48 48	9.8	66.0	290.0	110E15 46	07S45 22	8.9	66.0
120.0	110E24 58	07S49 40	9.9	66.0	300.0	110E16 24	07S44 46	8.3	66.0
130.0	110E24 25	07S50 25	9.8	66.0	310.0	110E17 04	07S44 18	7.8	66.0
140.0	110E24 11	07S51 34	11.1	66.0	320.0	110E17 43	07S43 56	7.4	66.0
150.0	110E23 17	07S52 06	10.9	66.0	330.0	110E18 25	07S43 46	6.9	66.0
160.0	110E22 32	07S53 04	11.9	66.0	340.0	110E19 08	07S43 48	6.3	66.0
170.0	110E21 27	07S53 24	12.0	66.0	350.0	110E19 45	07S43 51	5.9	66.0

Covered Area : 254.28 sqkm

A.2.8 DEPOK

User : sysadmin
Calculation Mode : Fieldstrength Contour GE 84
Calculation Model : ITU-R 1546 Database E50 Stereo
Receiver height : 10.0
Radio Service : FM Broadcast
Clear. Angle Corr.: Yes
Use Land-Sea Disc.: Yes
Calculated : 16/04/2008 23:12 CHIRplus_BC V 4.4.2

Wanted Transmit. : Depok
Frequency/MHz : 100.000 Chan. :
MaxERP kW / dBkW : 1.000 / 0.000
Longit. / Latit. : 110E24 52 / 07S46 07
Heff Max : 101 Country: INS
Polarisation : V OS :
Antenna Dir : ND Service: FM Broadcast
Offset : 0 System : 4

Fieldstrength Contour for Transmitter : Depok

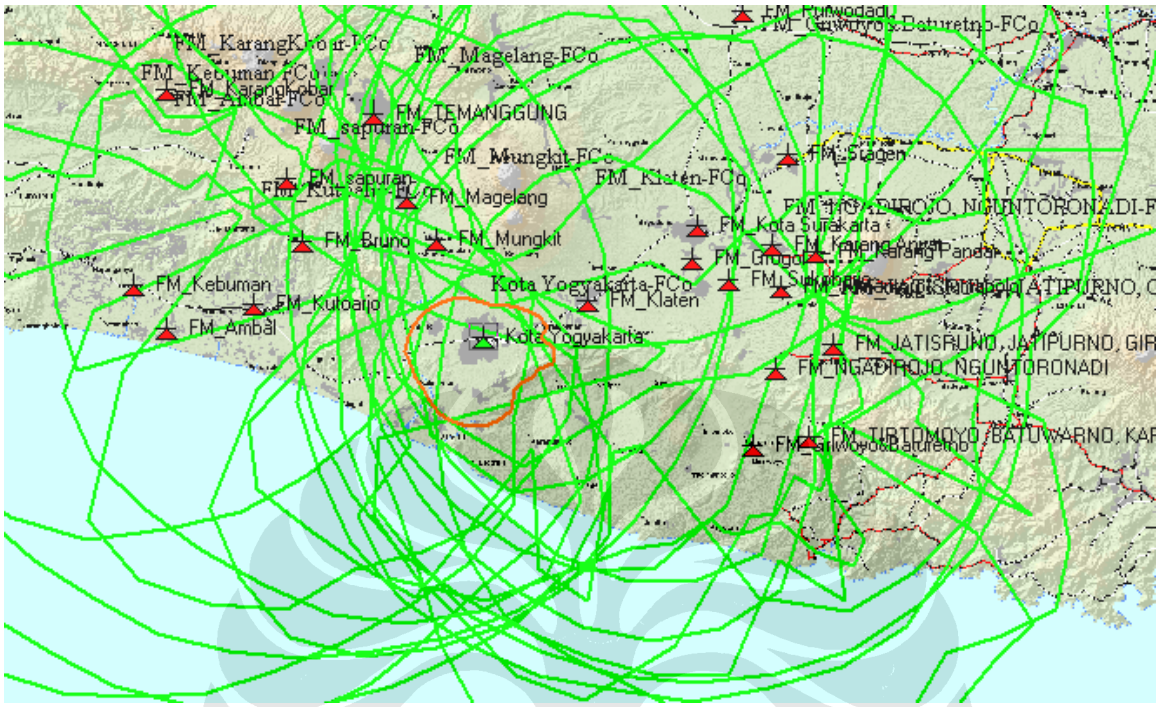
AZM	LONGITUDE	LATITUDE	DIST	VALUE	AZM	LONGITUDE	LATITUDE	DIST	VALUE
0.0	110E24 52	07S43 34	4.7	66.0	180.0	110E24 52	07S52 43	12.2	66.0
10.0	110E25 18	07S43 36	4.7	66.0	190.0	110E23 42	07S52 40	12.3	66.0
20.0	110E25 46	07S43 39	4.9	66.0	200.0	110E22 33	07S52 24	12.4	66.0
30.0	110E26 16	07S43 43	5.1	66.0	210.0	110E21 35	07S51 43	12.0	66.0
40.0	110E26 48	07S43 50	5.5	66.0	220.0	110E20 50	07S50 52	11.5	66.0
50.0	110E27 20	07S44 03	5.9	66.0	230.0	110E20 16	07S49 56	11.0	66.0
60.0	110E27 58	07S44 20	6.6	66.0	240.0	110E19 56	07S48 56	10.4	66.0
70.0	110E28 37	07S44 46	7.3	66.0	250.0	110E19 47	07S47 57	9.9	66.0
80.0	110E29 10	07S45 22	8.0	66.0	260.0	110E19 44	07S47 00	9.5	66.0
90.0	110E29 27	07S46 07	8.4	66.0	270.0	110E20 02	07S46 07	8.9	66.0
100.0	110E28 41	07S46 47	7.1	66.0	280.0	110E20 26	07S45 20	8.3	66.0
110.0	110E28 40	07S47 29	7.4	66.0	290.0	110E20 57	07S44 42	7.6	66.0
120.0	110E28 46	07S48 21	8.3	66.0	300.0	110E21 34	07S44 14	7.0	66.0
130.0	110E28 11	07S48 52	8.0	66.0	310.0	110E22 13	07S43 55	6.3	66.0
140.0	110E27 45	07S49 32	8.3	66.0	320.0	110E22 48	07S43 41	5.9	66.0
150.0	110E27 06	07S49 57	8.2	66.0	330.0	110E23 23	07S43 34	5.4	66.0
160.0	110E26 18	07S50 02	7.7	66.0	340.0	110E23 54	07S43 31	5.1	66.0
170.0	110E25 52	07S51 45	10.6	66.0	350.0	110E24 24	07S43 31	4.9	66.0

Covered Area : 216.14 sqkm

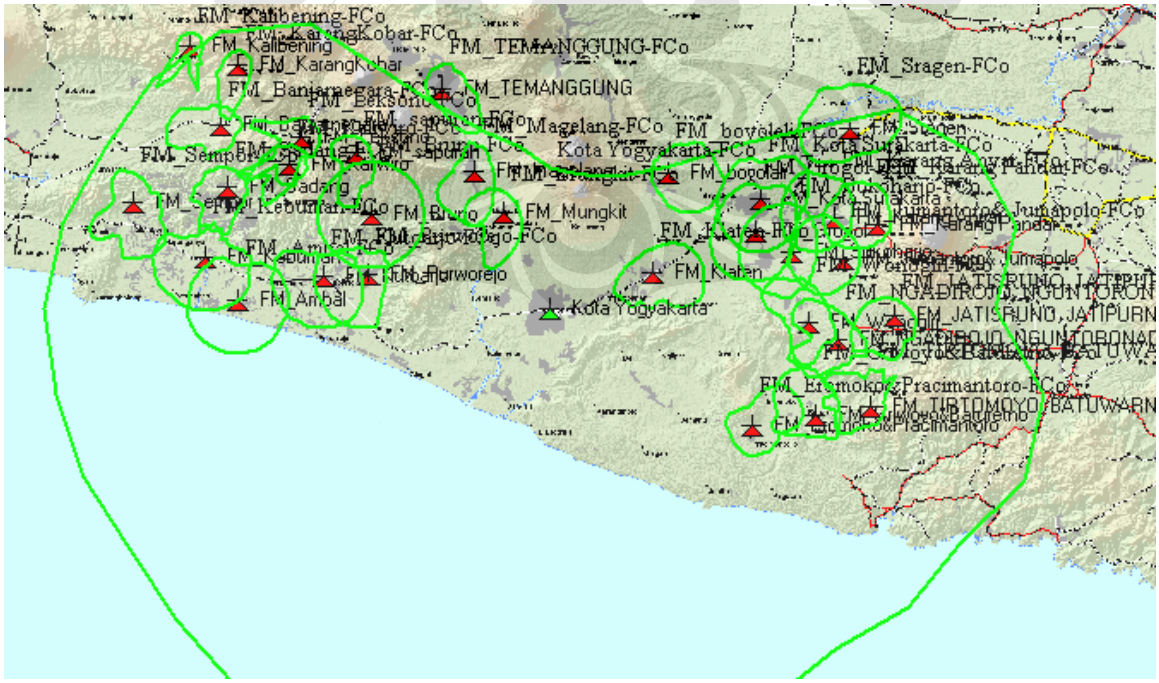
LAMPIRAN B

SIMULASI PENENTUAN DAERAH POTENSI INTERFERENSI

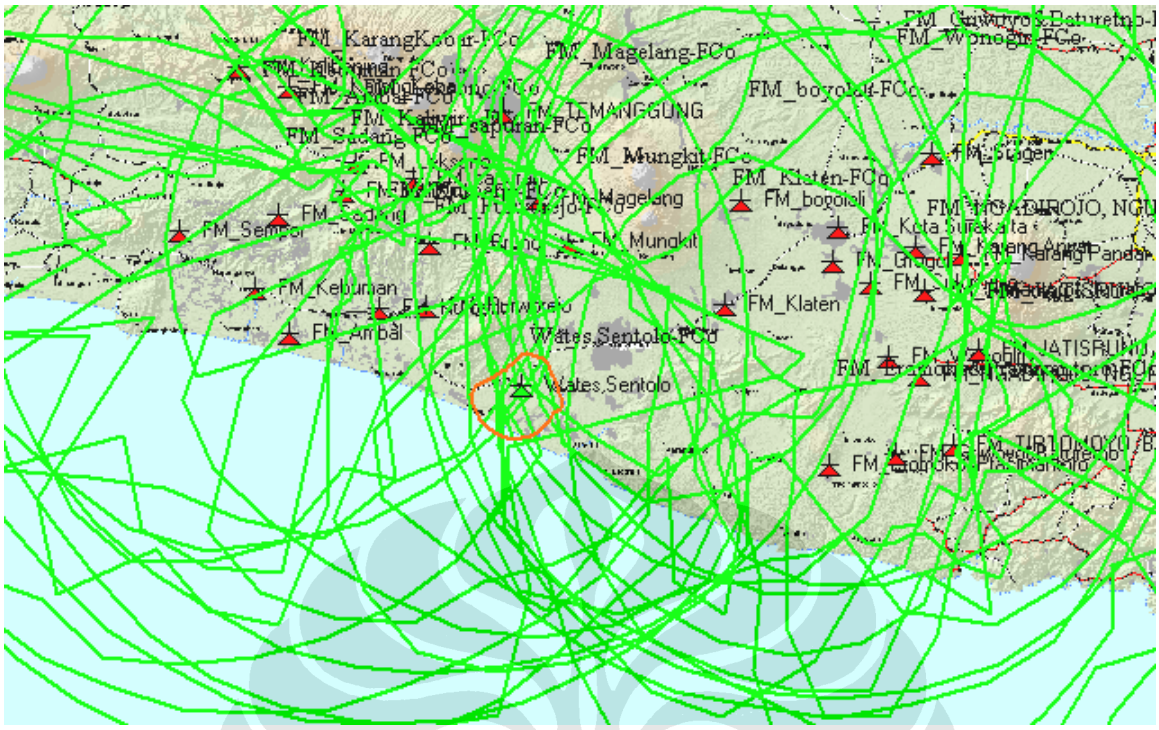
B.1 KOTA YOGYAKARTA SEBAGAI *WANTED*



B2. KOTA YOGYAKARTA SEBAGAI *UNWANTED*

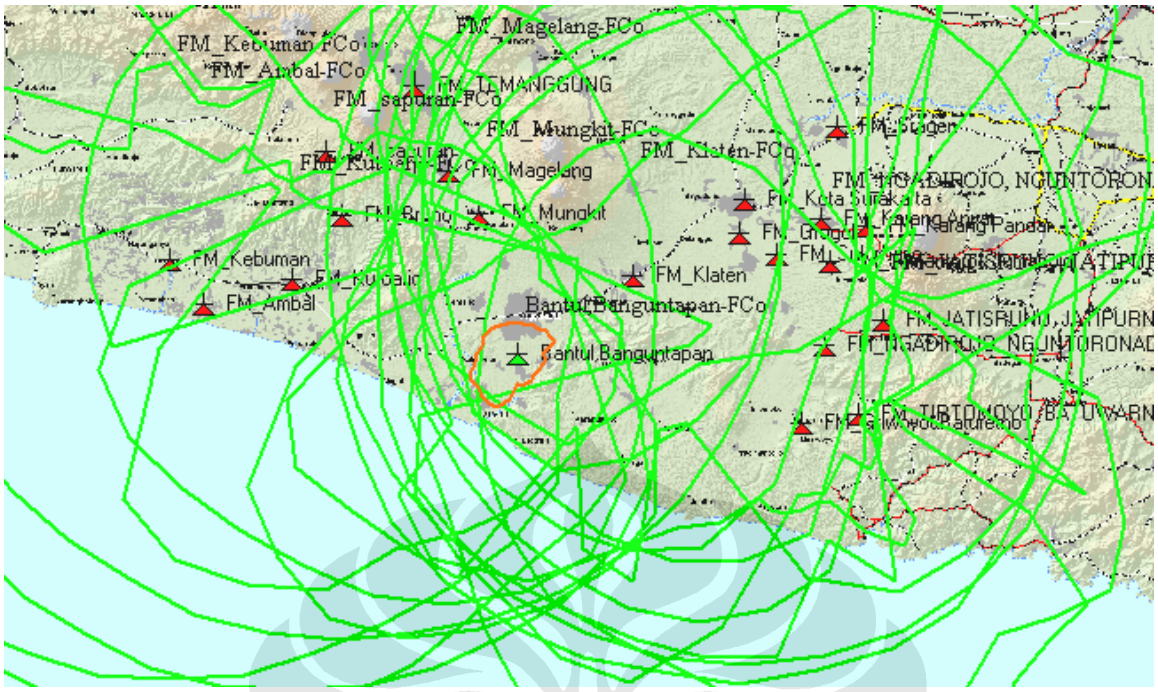


B3. WATES, SENTOLO SEBAGAI *WANTED*

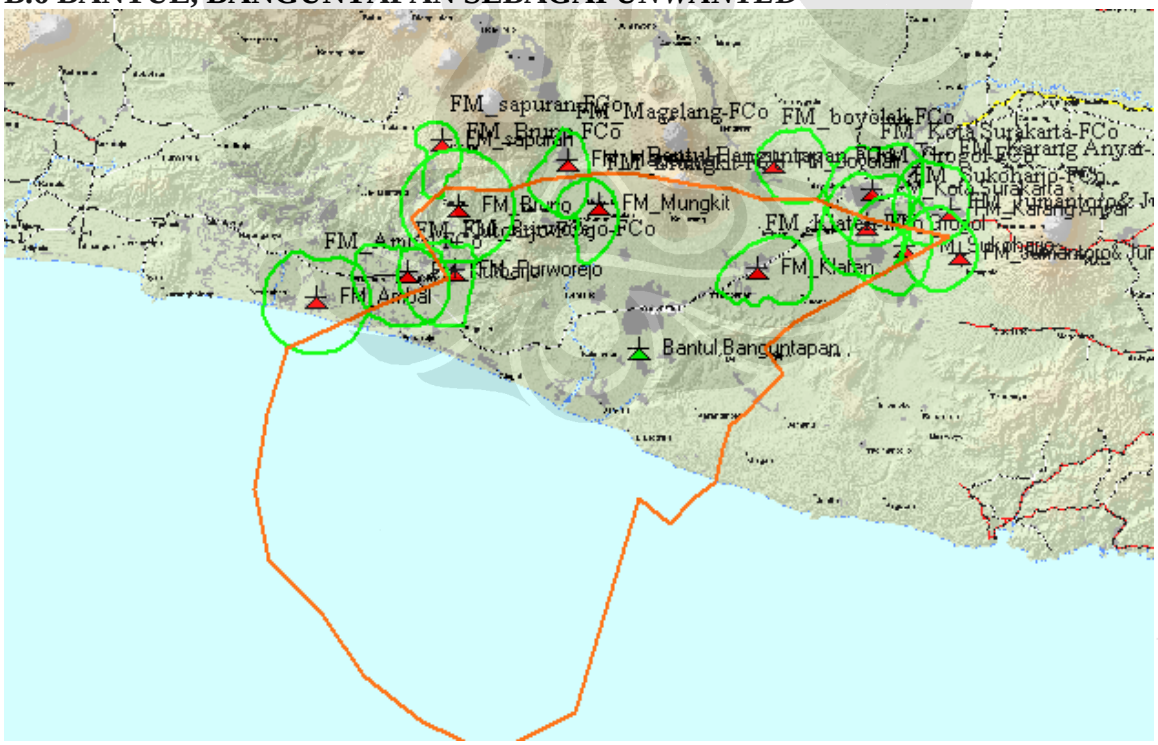


B.4 WATES, SENTOLO SEBAGAI *UNWANTED*

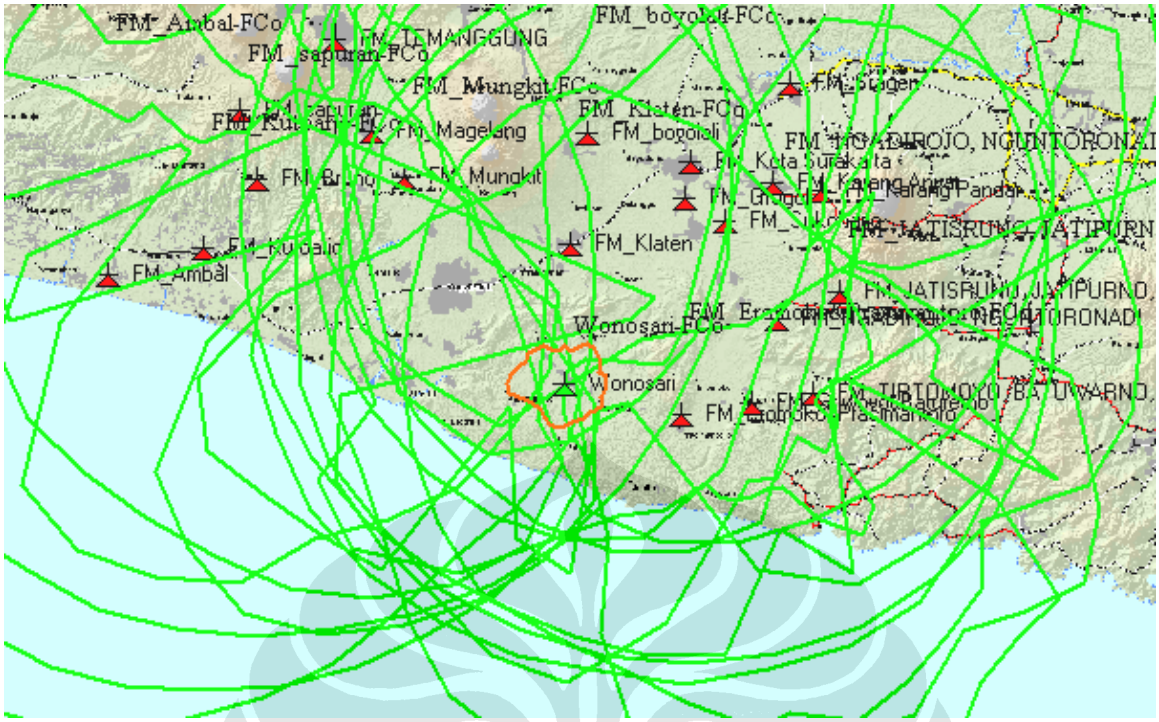
B.5 BANTUL, BANGUNTAPAN SEBAGAI *WANTED*



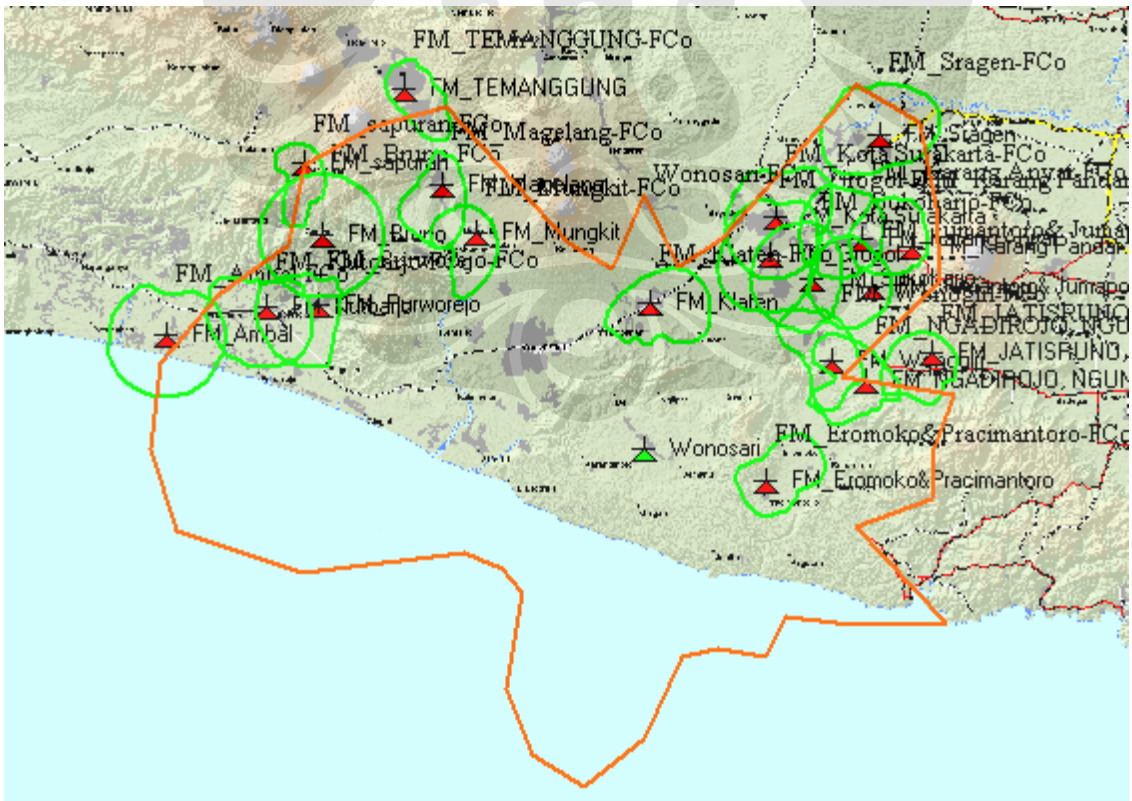
B.6 BANTUL, BANGUNTAPAN SEBAGAI *UNWANTED*



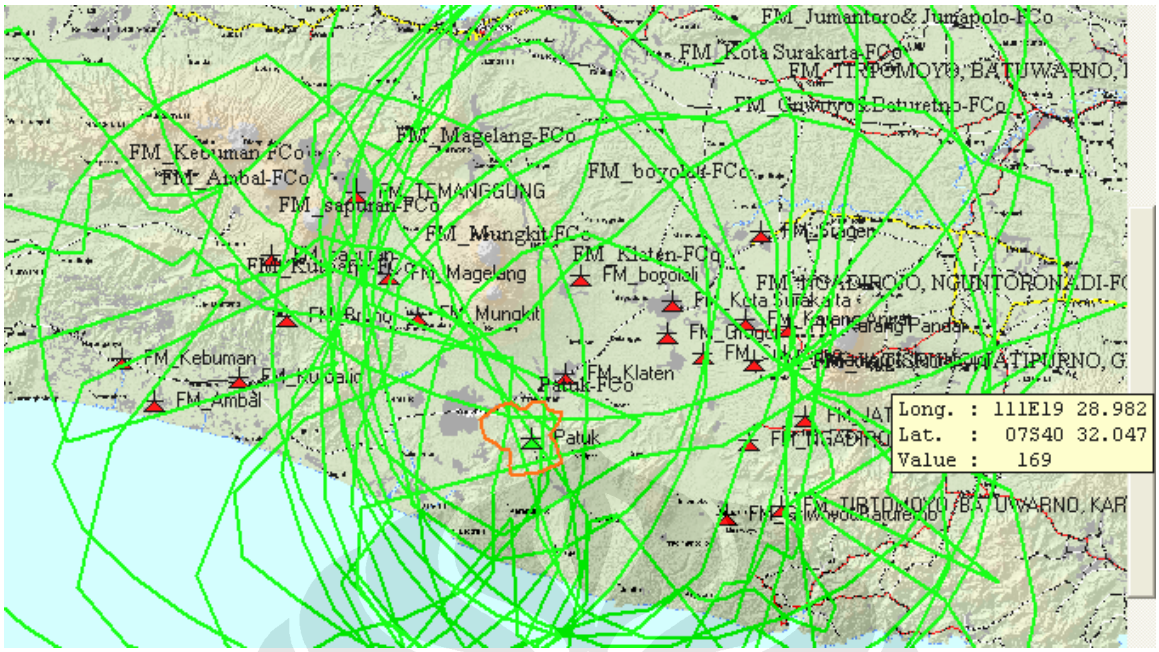
B.7 WONOSARI SEBAGAI *WANTED*



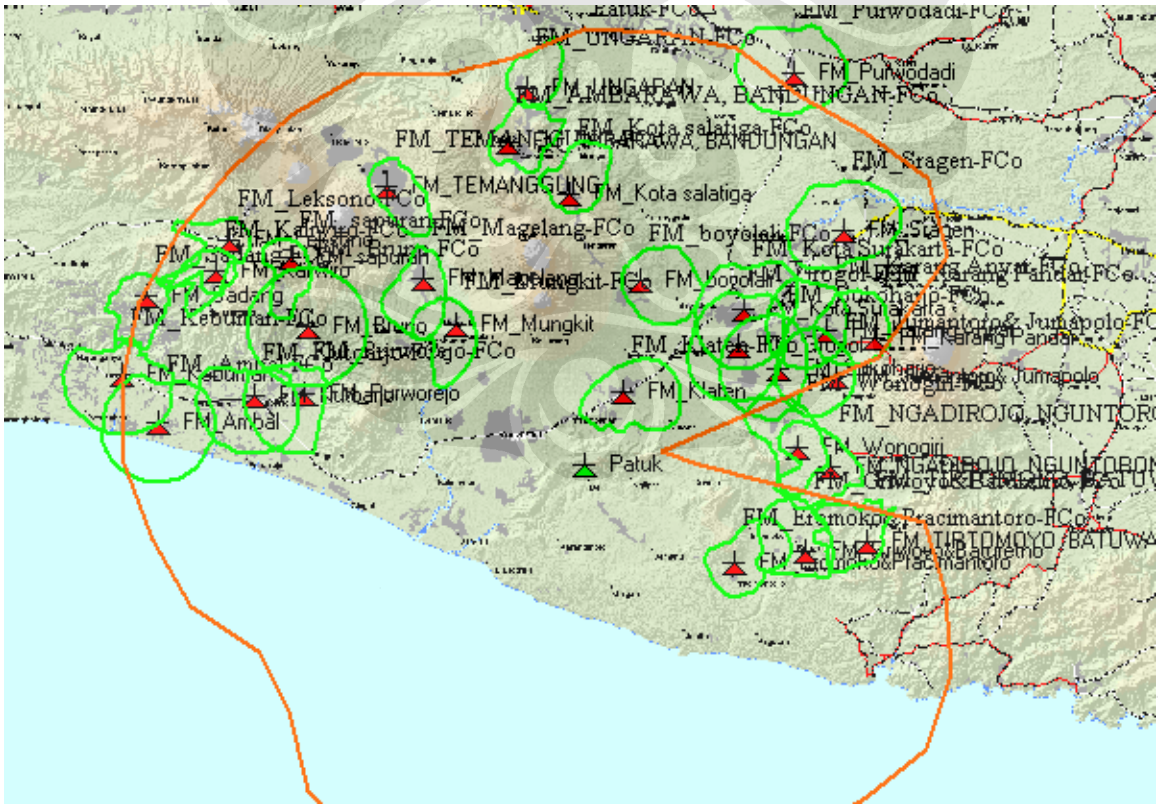
B.8 WONOSARI SEBAGAI *UNWANTED*



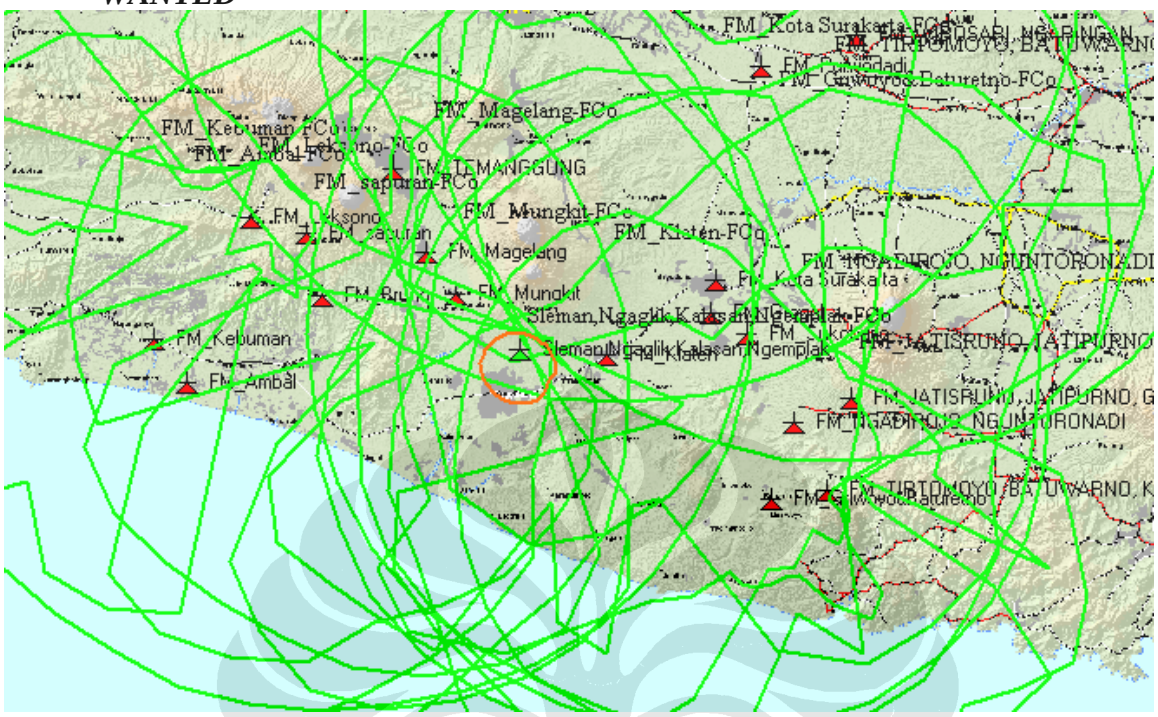
B.9 PATOK SEBAGAI *WANTED*



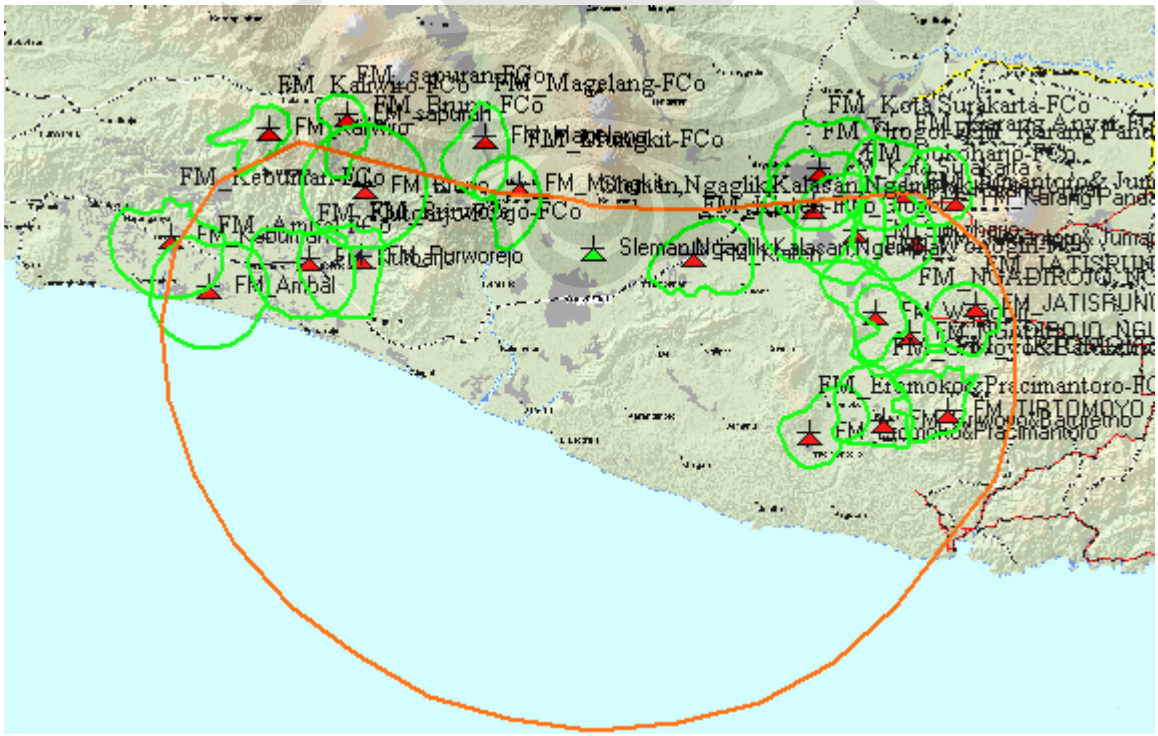
B.10 PATOK SEBAGAI *UNWANTED*



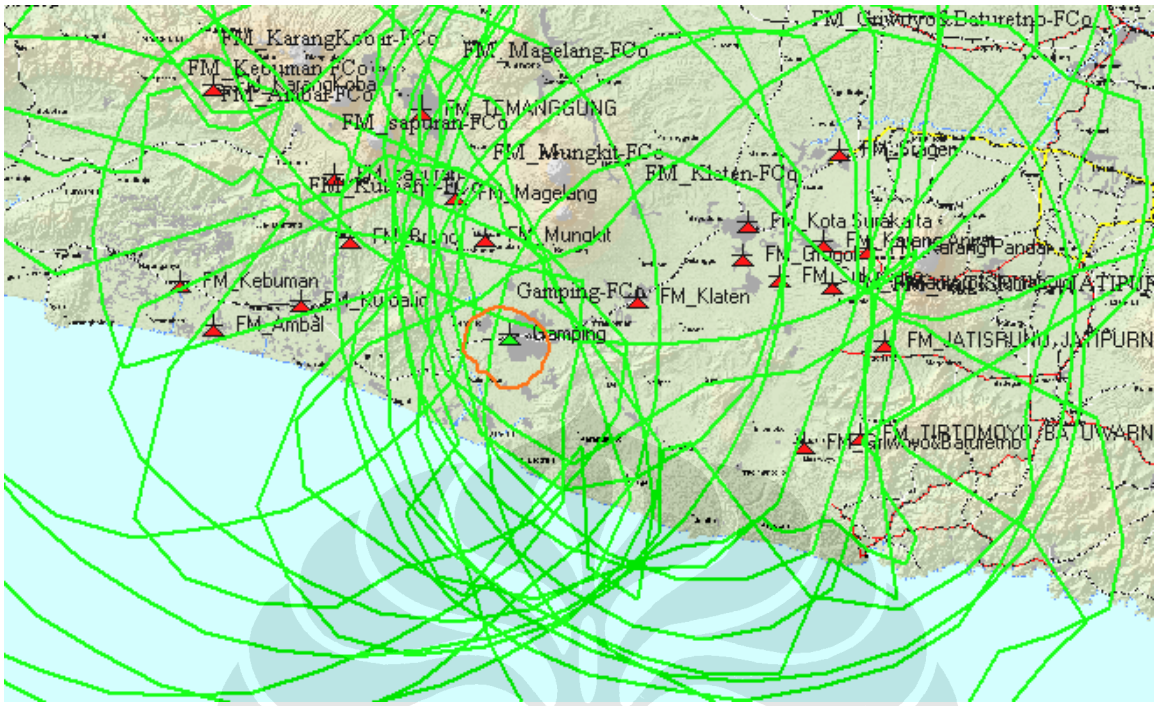
B.11 SLEMAN, NGAGLIK, KALASAN, NGEMPLAK, PAKEM SEBAGAI *WANTED*



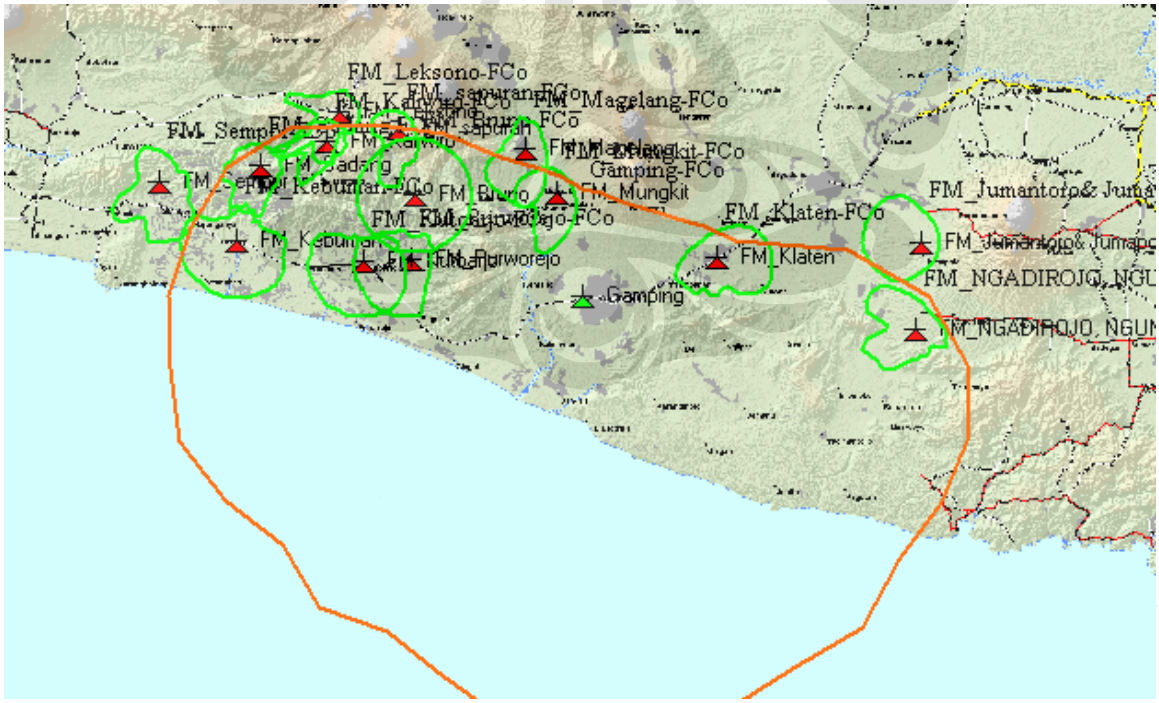
B.12 SLEMAN, NGAGLIK, KALASAN, NGEMPLAK, PAKEM SEBAGAI *UNWANTED*



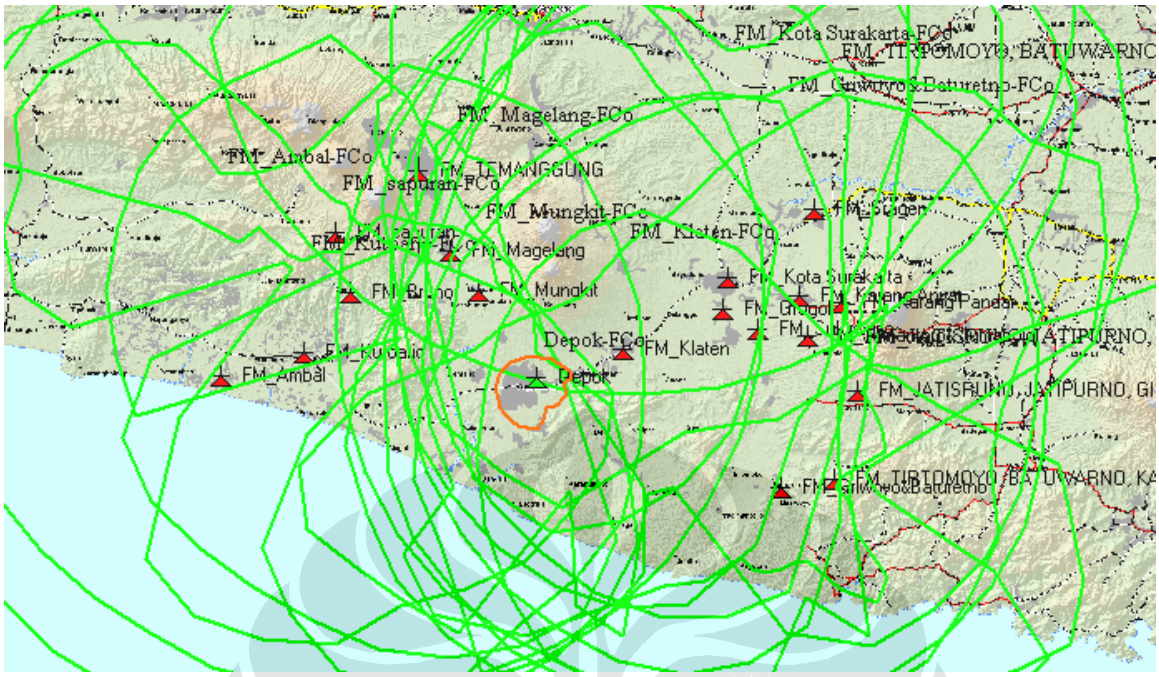
B.13 GAMPING SEBAGAI *WANTED*



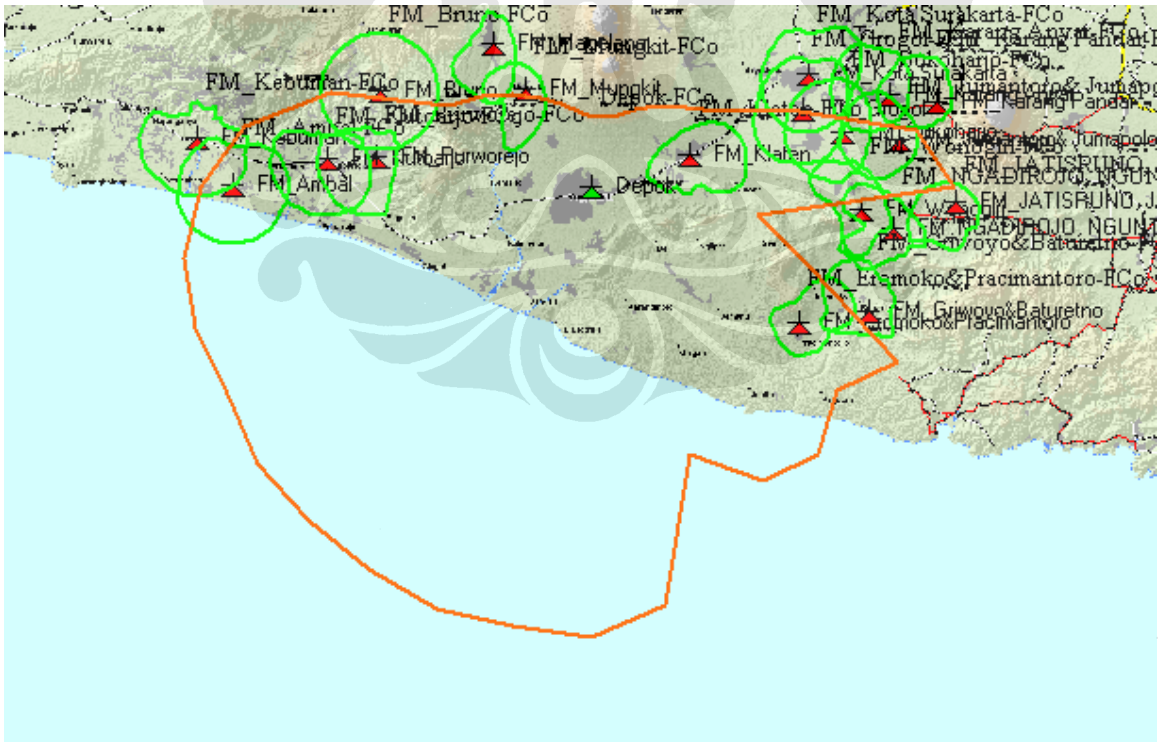
B.14 GAMPING SEBAGAI *UNWANTED*



B.15 DEPOK SEBAGAI *WANTED*



B.16 DEPOK SEBAGAI *UNWANTED*



LAMPIRAN C
C.1 Revisi KM 15 Propinsi DIY

I. PROPINSI D.I. YOGYAKARTA

A. RADIO SIARAN KELAS B

NO	WILAYAH	NOMOR KANAL
1	KOTA YOGYAKARTA	4, 12, 36, 52, 71, 79, 87, 103, 119, 138, 146, 154, 170, 186

B. RADIO SIARAN KELAS C

NO	WILAYAH		NOMOR KANAL
1.	KAB. KULON PROGO		
	a.	WATES, SENTOLO	63, 130, 197
2.	KAB. BANTUL		
	a.	BANTUL, BANGUNTAPAN	20, 67, 134, 201
3.	KAB. GUNUNG KIDUL		
	a.	WONOSARI	57, 124, 191
	b.	PATOK	111
4	KAB. SLEMAN		
	a.	BERAN	-
	b.	SLEMAN, NGAGLIK, KALASAN, NGEMPLAK, PAKEM	16, 32, 48, 83, 99, 115, 150, 166, 182
	c.	GAMPING	44
	d.	DEPOK	8, 28, 75, 95, 142, 162, 178

C.2 Revisi KM 15 Propinsi Jawa Tengah

II. PROPINSI JAWA TENGAH

A. RADIO SIARAN KELAS B

NO	WILAYAH	NOMOR KANAL
1	KOTA SEMARANG	3, 7, 11, 15, 19, 23, 27, 35, 43, 51, 59, 70, 74, 78, 82, 86, 90, 94, 102, 110, 114, 118, 126, 137, 141, 145, 149, 153, 157, 161, 169, 177, 185, 193

B. RADIO SIARAN KELAS C

NO	WILAYAH		NOMOR KANAL
1	KAB. CILACAP		
	a.	CILACAP	1, 17, 33, 68, 84, 100, 135, 151, 167
	b.	MAJENANG	67, 134
	c.	KAWUNGGATEN, JERUK LEGI	119, 186
	d.	KESUGIHAN, ADIPALA, MAOS	52
2	KAB. BANYUMAS		
	a.	PURWOKERTO, PATIK RAJA, KARANG LAWAS	40, 48, 56, 64, 107, 115, 123, 131, 174, 182, 190, 198
	b.	SOKARAJA, KALIBAGOR	44, 111, 178
	c.	WANGON, JATILAWANG , PURWOJATI	29, 96, 163
3	KAB. PURBALINGGA		
	a.	PURBALINGGA	5, 13, 21, 72, 80, 88, 139, 147, 155

4	KAB. BANJAR NEGARA		
	a.	BANJAR NEGARA, WANADADI, BANJARMANGU, MADUKARA	27, 35, 94, 102, 161, 169
	b.	KALI BENING, PUNGELAN	67, 134
	c.	WANAYASA, PEJAWARAN, PAGETAN, KARANG KOBAR	59, 126, 193
5	KAB. KEBUMEN		
	a.	KEBUMEN, ALIAN, KLIRONG	8, 24, 75, 91, 142, 158
	b.	KARANG ANYAR, GOMBONG, SEMPOR	31, 98, 165
	c.	KARANGGAYAM, SADANG	19, 86, 153
	d.	AMBAL, KUTOWINANGUN, MIRIT	42, 109, 176
6	KAB. PURWOREJO		
	a.	PURWOREJO, BANYU URIP	59, 126, 193
	b.	KUTOARJO, BAYAN	10, 77, 144
	c.	BRUNO, KEMIRI, PITURUH	66, 133
7	KAB. WONOSOBO		
	a.	WONOSOBO, LEKSONO, SELOMERTO, KALIKAJAR, KERTEK	38, 46, 105, 113, 172, 180
	b.	KALIWIRO, WADASLINTANG	50, 117, 184
	c.	SAPURAN	62, 129, 196

8	KAB. MAGELANG		
	a.	MUNGKID / MUNTILAN / SALAM	40, 107, 174
9	KAB. BOYOLALI		
	a.	BOYOLALI	61, 128, 195
10	KAB. KLATEN		
	a.	KLATEN	24, 91, 158
11	KAB. SUKOHARJO		
	a.	SUKOHARJO	50, 117, 184
	b.	GROGOL	34, 101, 168
12	KAB. WONOGORI		
	a.	WONOGIRI	65, 132
	b.	EROMOKO, PRACIMANTORO,	10, 77, 144
	c.	GIRIWOYO, BATURETNO,	18, 85, 152
	d.	TIRTOMOYO, BATUWARNO, KARANG TENGAH	41, 108, 175
	e.	NGADIROJO, NGUNTORONADI	57, 124, 191,
	f.	JATISRUNO, JATIPURNO,	26, 93, 160
13	KAB. KARANG ANYAR		
	a.	KARANG ANYAR	21, 88, 155
	b.	KARANG PANDAN	198
	c.	JUMANTONO, JUMAPOLO	1, 68, 135
14	KAB. SRAGEN		

	a.	SRAGEN	5, 13, 72, 80, 139, 147
15	KAB. GROBOGAN		
	a.	PURWODADI	40, 107
	b.	WIROSARI, NGARINGAN,	100, 167
16	KAB. BLORA		
	a.	BLORA	50, 58, 66, 117, 125, 133, 184, 192
	b.	KEDUNG TUBAN, KRADENAN, CEPU	45, 112, 179
17	KAB. REMBANG		
	a.	REMBANG	10, 26, 42, 77, 93, 109, 144, 160, 176
	b.	PANCUR, LASEM, PAMOTAN	30, 97, 164
	c.	KRAGAN, SEDAN, SARANG	102, 169
	d.	GUNEM, SALE	62, 129, 196
18	KAB. PATI		
	a.	PATI	1, 17, 33, 68, 84, 135, 151
	b.	BATANGAN, JUWANA, JAKENAN	21, 88, 155
	c.	GUNUNGWUNGKAL, CLUWAK, TAYU	13, 80, 147
19	KAB. KUDUS		
	a.	KUDUS	62, 129, 196

20	KAB. JEPARA		
	a.	JEPARA	67, 98, 134, 165
	b.	BANGSRI/KELING	5, 72, 139
21	KAB. DEMAK		
	a.	DEMAK	173
22	KAB. SEMARANG		
	a.	UNGARAN	47, 181
	b.	AMBARAWA, BANDUNGAN	30
23	KAB. TEMANGGUNG		
	a.	TEMANGGUNG	6, 22, 73, 89, 140, 156

24	KAB. KENDAL		
	a.	KENDAL	55, 122, 189
	b.	WELERI	64, 131, 198
25	KAB. BATANG		
	a.	BATANG	53, 120, 187
	b.	WONOTUNGGAL, BANDAR	33, 100, 167
	c.	BLADO, REBAN, BAWANG	40, 106, 174
26	KAB. PEKALONGAN		

	a.	KAJEN, KEDUNGWUNI	22, 89, 156
	b.	WIRADESA	14, 81, 148
	c.	DORO, LEBAKBARANG, TALUN	9, 76, 143
27	KAB. PEMALANG		
	a.	PEMALANG	2, 18, 69, 85, 136, 152
	b.	BANDARBOLANG, RANDUDONGKAL	61, 128, 195
28	KAB. TEGAL		
	a.	DESA BUMIJAWA, BOJONG	35, 51
	b.	SLAWI	102, 118, 169, 185
	c.	SURADADI, TARUB	6, 73, 140
29	KAB. BREBES		
	a.	BREBES	30, 46, 97, 113, 164, 180
	b.	BUMI AYU / PAGUYANGAN	9, 60, 76, 127, 143, 194
	c.	BANJARHARJO/ KETANGGUNGAN	6, 73, 140
30	KOTA MAGELANG		1, 55, 68, 122, 135, 189
31	KOTA SURAKARTA		38, 46, 54, 105, 113, 121, 172, 180, 188
32	KOTA SALATIGA		97, 164
33	KOTA PEKALONGAN		37, 96, 104, 163, 171
34	KOTA TEGAL		25, 41, 57, 65, 92, 108, 124, 132, 159, 175, 191