BAB III

DESAIN, RANCANGAN DAN IMPLEMENTASI PROGRAM *ZEPHEMERIS*

A. Metode Jean Meeus

1. Biografi Jean Meeus

Jean Meeus lahir pada tanggal 12 Desember 1928 M. Ia adalah seorang ahli Astronomi asal Belgia yang memfokuskan diri dalam mempelajari Mekanika langit, Matematika, dan Astronomi Bola. Jean Meeus mempelajari Matematika di University of Leuven, Belgia dan mendapatkan gelar sarjana pada tahun 1953. Pada tahun 1986, Jean Meeus memenangkan Amateur Achivement Award dari Astronomical Society of Pasific.[1] Jean Meeus juga merupakan Anggota Astronomical Society of France (ASF) sejak tahun 1948. Jean Meeus telah menerbitkan lebih dari 100 artikel dari ASF. Jean Meeus juga menjadi editor dalam Almanak perusahaannya selama 25 tahun. Salah satu temuan dari Jean Meeus adalah Asteroid 2213 Meeus. Hingga akhir hayatnya, Jean Meeus mengabdikan dirinya menjadi seorang *meteorologist* di Airport, Brussel.[2] Diantara karya-karyanya ialah:

a. Canon of Solar Eclipse (1966)

[1] *Amateur Achievement Award* adalah suatu penghargaan tahunan yang diberikan sejak 1976 M sebagai bentuk penghargaan terhadap para Astronom atau para Astronom amatir yang tidak bekerja di dunia Astronomi secara professional atas kontribusinya yang signifikan. Lihat https://www.astrosociety.org/about-us/awards/amateur-achievement-award-of-astronomical-, diakses pada tanggal 18 November 2016 pukul 07.58 WIB.

[2] Jean Meeus, *Mathematical Astronomy Morsels*, Virginia: Willman-Bell, Inc., 1997, hlm. iii.

b. Astronomical Formulae for Calculators (1979)
c. Astronomical Tables of the Sun, Moon, and Planets (1983)
d. Transits (1989)
e. Elements of Solar Eclipse 1951-2200 (1989)
f. Astronomical Algorithms (1991)
g. Mathematical Astronomy Morsels (1997)
h. More Mathematical Astronomy Morsels (2002)
i. Mathematical Astronomy Morsels (2004)
j. Mathematical Astronomy Morsels (2007)
k. Mathematical Astronomy Morsels (2009)

2. Algoritma Perhitungan Data *Ephemeris* Matahari dan Bulan metode Jean Meeus

Aplikasi ini menggunakan algoritma Jean Meeus dalam proses perhitungannya, sebagaimana dijelaskan sebelumnya. Adapun alur proses perhitungan algoritma secara lengkap adalah sebagai berikut:

a. Algoritma Perhitungan *Julian Day*, Selisih antara *Universal Time* dan *Dynamical Time* (ΔT), *Julian Day Ephemeris*, *Julian Centuries* (T) dan *Julian Millenia* (ţ)

1) Perhitungan *Julian Day*

Input data yang diperlukan untuk melakukan perhitungan data *ephemeris* yakni data tanggal dan waktu. Data tanggal berisi tanggal, bulan dan tahun, data waktu berisi jam, menit dan detik. Selanjutnya, data tanggal terlebih dahulu

diubah ke dalam sistem Julian, sedangkan data waktu diubah ke dalam UT atau GMT (misal 08.00 WIB = 01.00 GMT). Berikut ini algoritma untuk mengubah tanggal dan tahun ke sistem *Julian Day* (JD):[3]

Jika bulan > 2, maka M = bulan, dan Y = tahun

Jika bulan = 1 atau 2, maka M = bulan+12, dan Y = tahun-1

A = INT (Y/100)

B = 2-A+INT(A/4)

JD = INT (365,25 * (Y + 4716)) + INT (30,6001 * (M+1)) + B + tanggal + (jam + menit/60 + detik/3600) /24 - 1524,5

2) Algoritma Perhitungan ΔT

Langkah selanjutnya adalah menghitung ΔT. Terdapat banyak rumus yang diperkenalkan oleh Jean Meeus untuk mencari ΔT. Sebagaimana diterangkan pada bab sebelumnya, nilai ΔT dari tahun ke tahun senantiasa berubah, sehingga beberapa rumus perhitungan ΔT yang terdapat pada buku *Astronomical Algorithms* milik Jean Meeus, kadang menghasilkan data *error* untuk perhitungan ΔT pada tahun tertentu.[4] Adapun rumus yang terbaru yang Meeus perkenalkan

[3] Jean Meeus, *Astronomical Algorithms*, First Edition, Virginia: Willman-Bell, Inc., 1991, hlm. 61

[4] *Ibid*. hlm. 71-75

melalui *website* NASA yakni rumus logika ***polynomial expression for*** ΔT.[5]

Rumus *polynomial expression for* ΔT merupakan rumus yang disusun oleh Jean Meeus dan Fred Espenak. Rumus ini disusun untuk mengatasi masalah *error* hasil perhitungan pada beberapa rumus-rumus sebelum. Adapun cangkupan perhitungan dari rumus ini adalah dari tahun -1999 s/d +3000, atau sekitar lima millenium.[6] Berikut ini adalah logika perhitungan *polynomial expression for* ΔT:[7]

Jika tahun ≤ -500, maka:

ΔT = -20 + 32 * u^2

Di mana u = (y-1820)/100

Jika tahun diantara -500 s/d +500 maka:

ΔT = 10583.6 - 1014.41 * u + 33.78311 * u^2 - 5.952053 * u^3 - 0.1798452 * u^4 + 0.022174192 * u^5 + 0.0090316521 * u^6

Di mana u = y/100

Jika tahun diantara +500 s/d +1600, maka:

ΔT = 1574.2 - 556.01 * u + 71.23472 * u^2 + 0.319781 * u^3 - 0.8503463 * u^4 - 0.005050998 * u^5 + 0.0083572073 * u^6

Di mana u = (y-1000)/100

[5] http://eclipse.gsfc.nasa.gov/SEcat5/deltatpoly.html, diakses pada tanggal 18 November 2016 pukul 15.24 WIB.

[6] *Ibid.*

[7] *Ibid.*

Jika tahun diantara +1600 s/d +1700, maka:

ΔT = 120 - 0.9808 * t - 0.01532 * t^2 + t^3 / 7129

Di mana t = y - 1600

Jika tahun diantara +1700 s/d +1800, maka:

ΔT = 8.83 + 0.1603 * t - 0.0059285 * t^2 + 0.00013336 * t^3 - t^4 / 1174000

Di mana t = y - 1700

Jika tahun diantara +1800 s/d +1860, maka:

ΔT = 13.72 - 0.332447 * t + 0.0068612 * t^2 + 0.0041116 * t^3 - 0.00037436 * t^4 + 0.0000121272 * t^5 - 0.0000001699 * t^6 + 0.000000000875 * t^7

Di mana t = y - 1800

Jika tahun diantara 1860 s/d 1900, maka:

ΔT = 7.62 + 0.5737 * t - 0.251754 * t^2 + 0.01680668 * t^3-0.0004473624 * t^4 + t^5 / 233174

Di mana t = y - 1860

Jika tahun di antara 1900 s/d 1920, maka:

ΔT = -2.79 + 1.494119 * t - 0.0598939 * t^2 + 0.0061966 * t^3 - 0.000197 * t^4

Di mana t = y - 1900

Jika tahun di antara 1920 s/d 1941, maka:

ΔT = 21.20 + 0.84493*t - 0.076100 * t^2 + 0.0020936 * t^3

Di mana t = y - 1920

Jika tahun di antara 1941 s/d 1961, maka:

ΔT = 29.07 + 0.407*t - t^2/233 + t^3 / 2547

Di mana t = y - 1950

Jika tahun di antara 1961 s/d 1986, maka:

ΔT = 45.45 + 1.067*t - t^2/260 - t^3 / 718

Di mana t = y - 1975

Jika tahun di antara 1986 s/d 2005, maka:

ΔT = 63.86 + 0.3345 * t - 0.060374 * t^2 + 0.0017275 * t^3 + 0.000651814 * t^4 + 0.00002373599 * t^5

Di mana t = y - 2000

Jika tahun di antara 2005 s/d 2050, maka:

ΔT = 62.92 + 0.32217 * t + 0.005589 * t^2

Di mana t = y - 2000

Jika tahun diantara 2050 s/d 2150, maka:

ΔT = -20 + 32 * ((y-1820)/100)^2 - 0.5628 * (2150 - y)

Jika tahun ≥ 2150, maka:

ΔT = -20 + 32 * u^2

Di mana u = (y-1820)/100

3) Menghitung *Julian Day Ephemeris*, *Julian Centuries* (T), dan *Julian Millenia* (Ƭ)[8]

Langkah selanjutnya adalah menghitung JDE, T dan Ƭ, berikut rumus yang dipergunakan:

[8] Jean Meeus, *loc.cit.*

JDE = JD + ΔT

T (TD) = (JDE – 2452545) / 36525

ʯ = T / 10

b. Algoritma Perhitungan Data *Ephemeris* Matahari

1) Koreksi Posisi Planet Bumi

Untuk mengetahui data *ephemeris* Matahari, sebelumnya dilakukan perhitungan posisi orbit planet Bumi dari Matahari. Terdapat 3 data posisi untuk masing-masing planet, yakni: L untuk bujur heliosentris planet, B untuk lintang heliosentris planet dan R untuk jarak planet-Matahari.[9]

Untuk perhitungan data posisi planet Bumi terdapat 6 kelompok koreksi untuk data L Bumi, 2 kelompok koreksi untuk data B Bumi dan 5 kelompok koreksi untuk data R Bumi. Data tabel koreksi *Periodic Terms* untuk planet Bumi dapat dilihat pada lampiran I. Adapun untuk menggunakan data koreksi pada tabel tersebut menggunakan rumus:

A * Cos(B + C * ʯ)

Koreksi pertama yakni koreksi bujur heliosentris Bumi (L), terdapat 6 kelompok koreksi pada tahap ini, yakni L0, L1, L2, L3, L4 dan L5. Di mana pada masing-masing kelompok koreksi terdapat suku-suku koreksi dengan jumlah yang berbeda-beda. Masing-masing suku koreksi tersebut dihitung

[9] *Ibid.* hlm. 205

menggunakan rumus yang telah disebutkan di atas. Selanjutnya suku-suku koreksi tersebut dijumlahkan pada masing-masing kelompok koreksi.

Terakhir, untuk mendapatkan bujur heliosentris Bumi digunakan rumus:

$$L = (L0 + L1 * \tau + L2 * \tau^2 + L3 * \tau^3 + L4 * \tau^4 + L5 * \tau^5) / 10^8$$

L merupakan bujur ekliptika Bumi diukur dari Matahari (heliosentris). Adapun untuk mengetahui bujur ekliptika Matahari dari Bumi (geosentris) menggunakan rumus berikut:

$\Theta = L + 180$, hasil dalam derajat.

Setelah itu, menghitung koreksi untuk Θ dengan rumus:

$\Theta^t = \Theta - 0{,}09033$", hasil dalam derajat.

Hasilnya merupakan *true geometric longitude* matahari.

Koreksi kedua adalah koreksi lintang heliosentris Matahari (B), terdapat 2 kelompok koreksi pada tahap ini, yakni B0 dan B1. Proses koreksinya sama dengan koreksi L. Selanjutnya untuk mengetahui B menggunakan rumus berikut:

$B = ((B0 + B1 * \tau) / 10^8) * 1$, hasil dalam bentuk radian, kemudian dikonversi ke dalam detik busur.

Setelah itu menghitung koreksi B dengan rumus:

$$\lambda' = \Theta - 1{,}397 * T - 0{,}00031 * T^2$$

$\Delta B = 0{,}03916 * (COS\ \lambda' - SIN\ \lambda')$, di mana untuk pemrograman java, λ' dirubah ke dalam bentuk desimal terlebih dahulu.

Selanjutnya untuk mengetahui *apparent latitude* Matahari menggunakan rumus:

$$ß = B + \Delta B$$

Koreksi ketiga adalah koreksi jarak Matahari-Bumi (R). Terdapat 5 kelompok koreksi. Adapun proses koreksinya sama dengan koreksi L dan B. Selanjutnya untuk mengetahui *true geocentric distance* menggunakan rumus:

$$R = (R0 + R1 * Ţ + R2 * Ţ^2 + R3 * Ţ^3 + R4 * Ţ^4) / 10^8$$

2) Algoritma Perhitungan *Nutasi* dan *True Obliquity*

Langkah pertama yakni menghitung *mean obliquity* dengan rumus:

$E_o = 23^o\ 26'\ 21{,}448'' + (-4680{,}93 * U - 1{,}55 * U^2 + 1999{,}25 * U^3 - 51{,}38 * U^4 - 249{,}67 * U^5 - 39{,}05 * U^6 + 7{,}12 * U^7 + 27{,}87 * U^8 + 5{,}79 * U^9 + 2{,}45 * U^{10})/3600$, hasil dalam derajat.

Di mana $U = T/100$

Untuk mengetahui *true obliquity* dan *nutasi*, diperlukan perhitungan koreksi $\Delta\varepsilon$ dan $\Delta\psi$ menggunakan tabel *terms of the 1980 IAU Theory of Nutations*, sebagaimana terdapat pada lampiran II dan III. Namun sebelum menghitung $\Delta\psi$ dan $\Delta\varepsilon$ dengan tabel tersebut, diperlukan perhitungan *multiple arguments* terlebih dahulu. Berikut rumus perhitungan *multiple arguments* untuk $\Delta\psi$ dan $\Delta\varepsilon$:

D = $297{,}85036 + 445267{,}11148*T - 0{,}0019142*T^2 + T^3/189474$, hasil dalam derajat.

M_o = $357{,}52772 + 35999{,}05034*T - 0{,}0001603*T^2 - T^3/300000$, hasil dalam derajat.

M^c = $134{,}96298 + 477198{,}867398*T + 0{,}0086972*T^2 + T^3/56250$, hasil dalam derajat.

F = $93{,}27191 + 483202{,}017538*T - 0{,}0036825*T^2 + T^3/327270$, hasil dalam derajat.

Ω^c = $125{,}04452 - 1934{,}136261*T + 0{,}0020708*T^2 + T^3/450000$, hasil dalam derajat.

Langkah selanjutnya adalah menghitung koreksi $\Delta\psi$ pada tabel yang terdapat pada lampiran II menggunakan rumus:

(Coefficient 1 + Coefficient 2 * T) * sin (Multiple Arguments)

Setelah itu seluruh koreksi dijumlahkan, kemudian digunakan untuk menghitung *Nutasi* ($\Delta\psi$).

$\Delta\psi$ = jumlah koreksi/10000/3600, hasil dalam derajat.

Selanjutnya algoritma perhitungan berpindah ke tabel koreksi Δε pada lampiran III. Koreksi dihitung menggunakan rumus:

(Coefficient 1 + Coefficient 2 * T) * cos (Multiple Arguments)

Terakhir, yakni menghitung *apparent longitude* Matahari berdasarkan hasil perhitungan di atas dengan algoritma:

$\lambda = \Theta^t + \Delta\psi$ + Aberasi, hasil dalam bentuk derajat.

3) Menghitung *Right Ascenssion* dan *Apparent Declination* (Deklinasi Matahari)[10]

Rumus *apparent right ascension*:

α = Atan ((Sin λ * Cos ε – Tan β *Sin ε) / Cos λ), hasil dalam derajat.

Rumus *apparent declination*:

Sin δ = Sin β * Cos ε + Cos β * Sin ε * Sin λ, hasil dalam derajat.

4) Algoritma Perhitungan *Equation of Time*[11]

Sebelum menghitung *equation of time*, terlebih dahulu dilakukan perhitungan bujur rata-rata Matahari menggunakan rumus:

L_o = 280,4664567 + 360007,6982779 * τ - 0,03032028 * τ^2 + τ^3 /49931 - τ^4 / 15299 - τ^5 /

[10] *Ibid.* hlm. 89
[11] *Ibid.* hlm. 171-175

1988000, hasil dalam derajat dan di *modulus*-kan.

Rumus perhitungan *equation of time*:

Eq = L_o - 0,0057183 – α + Δψ *COS ε, hasil dalam menit waktu.

5) Algoritma Perhitungan Semi Diameter Matahari[12]

Sd = 15’ 59,63” / R , hasil dalam derajat.

c. Algoritma Perhitungan Data *Ephemeris* Bulan

1) Perhitungan Koreksi Posisi Bulan[13]

Langkah pertama dalam perhitungan posisi Bulan yakni menghitung *Moon’s mean longitude* terlebih dahulu menggunakan rumus berikut:

L’ = 218,3164591 + 481267,88134236 * T - 0,0013268*T^2 + T^3/538841 – T^4/65194000

Selanjutnya dilanjutkan pada perhitungan rumus-rumus *argument* berikut:

Rumus perhitungan *mean elongation of the Moon*:

D = 297,8502042 + 445267,1115168*T - 0,00163*T^2 + T^3/545868 – T^4/113065000

Rumus perhitungan *Sun’s mean Anomaly:*

M = 357,5291092 + 35999,0502909*T - 0,0001536*T^2 + T^3/24490000

Rumus perhitungan *Moon’s mean Anomaly*:

[12] *Ibid.* hlm. 359-361
[13] *Ibid.* hlm. 307-308

M’ = 134,9634114 + 477198,8676313*T + 0,008997*T^2 + T^3/69699 – T^4/14712000

Rumus *Argumen Bujur Bulan* (jarak rata-rata Bulan yang dihitung dari titik perpotongan lintasan bulan dengan *ekliptika*):

F = 93,2720993 + 483202,0175273*T - 0,0034029*T^2 – T^3/3526000 + T^4/863310000

Setelah itu dilanjutnya pada perhitungan 3 argumen tambahan dengan rumus sebagai berikut:

A1 = 119,75 + 131,849*T
A2 = 53,09 + 479264,29*T
A3 = 313,45 + 481266,484*T

Hasil perhitungan diatas dalam bentuk derajat dengan nilai yang belum berbatas, untuk mengubahnya menjadi nilai diantara 0-360°, hasil perhitungan terlebih dahulu di-*modulus*-kan. Adapun untuk digunakan dalam perhitungan pemrogaman *java* nilai derajat diubah menjadi desimal.

Langkah selanjutnya untuk menghitung posisi bulan yakni menghitung koreksi periodik berdasarkan tabel koreksi periodik yang terdapat pada lampiran IV untuk koreksi bujur Bulan, lampiran V untuk koreksi lintang Bulan dan lampiran VI untuk koreksi jarak Bumi-Bulan. Rumus yang digunakan

untuk menghitung koreksi pada tabel koreksi periodik bujur dan lintang Bulan adalah sebagai berikut:

Koreksi = Coefficient * Sin (Multiple Arguments)

Namun Jika nilai M pada tabel ≠ 0 (0 < M atau M > 0) maka:

Koreksi = Coefficient * E* Sin (Multiple Arguments)

Di mana E adalah eksentrisitas orbit Bumi

dan E = 1-0,002516 * T -0,0000074 * T^2

Sedangkan untuk menghitung koreksi pada tabel koreksi periodik jarak Bumi-Bulan menggunakan rumus:

Koreksi = Coefficient * Cos (Multiple Arguments)

Namun Jika nilai M pada tabel ≠ 0 (0 < M atau M > 0) maka:

Koreksi = Coefficient * E* Cos (Multiple Arguments)

Di mana E adalah eksentrisitas orbit Bumi

dan E = 1-0,002516 * T -0,0000074 * T^2

Selanjutnya seluruh koreksi dari masing-masing tabel dijumlah, di mana hasil penjumlahan tersebut yakni Σl untuk hasil koreksi periodik bujur Bulan, Σb untuk hasil koreksi periodik lintang Bulan dan Σr untuk koreksi periodik jarak Bumi-Bulan. Untuk mendapatkan hasil koreksi tambahan bujur Bulan menggunakan rumus:[14]

[14] *Ibid.* hlm. 312

Kor. tambahan Σl = (Σ l + 3958 * Sin (A1) + 1962 * SIN (L'-F) + 318 * Sin (A2))/1.000.000

Untuk mendapatkan hasil koreksi tambahan lintang Bulan menggunakan rumus:[15]

Kor. tambahan Σb = (Σb - 2235 * Sin (L') + 382 * Sin (A3) + 175* Sin (A1 - F) + 175 * Sin (A1 + F) + 127 * Sin (L' – M') - 115*Sin (L' + M')) / 1.000.000

Untuk menghitung koreksi tambahan untuk jarak Bumi-Bulan menggunakan rumus:[16]

Kor. tambahan Σr = Σr/1.000, hasil dalam satuan kilometer.

2) Perhitungan *True Longitude*, *True Latitude*, dan *True Geocentric Distance of The Moon*

Rumus yang digunakan untuk menghitung koordinat Bulan (*true longitude*, *latitude* dan *true geocentric distance of the Moon*) yakni: [17]

λ = L' (dalam format derajat) + kor. tambahan Σl, hasil dalam derajat.
β = kor. tambahan Σb, hasil dalam derajat
TGD = 385000,56 + kor. tambahan Σr, hasil dalam kilometer.

3) *Nutasi* dan *True Obliquity*

[15] *Ibid.*
[16] *Ibid.*
[17] *Ibid.*

Perhitungan *nutasi* dan *true obliquity* pada perhitungan data Bulan adalah sama dengan perhitungan *nutasi* dan *true obliquity* pada perhitungan data Matahari.

4) *Apparent Longitude* Bulan

Rumus yang digunakan yakni:[18]

Apparent Longitude = True Longitude + $\Delta\psi$

5) *Apparent Right Ascension* dan *Apparent Declination* (Deklinasi Bulan)

Rumus *apparent right ascension*:[19]

$$\alpha = \text{Atan}((\text{Sin}\,\lambda * \text{Cos}\,\varepsilon - \text{Tan}\,\beta * \text{Sin}\,\varepsilon) / \text{Cos}\,\lambda)$$

Rumus *apparent declination*:[20]

$$\text{Sin}\,\delta = \text{Sin}\,\beta * \text{Cos}\,\varepsilon + \text{Cos}\,\beta * \text{Sin}\,\varepsilon * \text{Sin}\,\lambda$$

6) Perhitungan *Horizontal Parallax* Bulan

Untuk menghitung *horizontal parallaks* Bulan, dapat menggunakan rumus berikut:[21]

Sin HP_c = 6378,14 / TGD bulan, hasil dalam satuan derajat.

7) Menghitung Semi Diameter Bulan

[18] *Ibid.* hlm. 313.
[19] *Ibid.*
[20] *Ibid.*
[21] *Ibid.* hlm. 263.

Ada beberapa rumus yang bisa digunakan untuk menghitung semi diameter Bulan, salah satunya yakni menggunakan rumus:[22]

Sd_c = 358473400 / Jarak Bumi-Bulan

8) Menghitung *Fraction Ilumination* Bulan

Untuk menghitung *fraction ilumination* Bulan (FIB), diperlukan data Matahari, sebab nilai FIB tergantung dari besar-kecilnya sudut elongasi Bulan. Data Matahari yang dibutuhkan untuk perhitungan FIB yakni α_o, δ_o dan R (*deklinasi*, *right ascension* dan jarak Bumi-Matahari). Adapun untuk langkah pertama yang dilakukan yakni dengan menghitung sudut *elongasi* Bulan terlebih dahulu dengan rumus:

$$\text{Cos el} = \text{Sin } \delta_o * \text{Sin } \delta_c + \text{Cos } \delta_o * \text{Cos } \delta_c * \text{Cos } (\alpha_o - \alpha_c)$$

Langkah selanjutnya yakni menghitung *phase angle of the Moon*. Namun Sebelum menghitung *phase angle*, terlebih dahulu nilai R diubah dari satuan AU ke dalam satuan kilometer, di mana 1 AU= 149.597.871 km. Adapun rumus untuk menghitung *phase angle of the Moon* yakni:

Tan i = (R * Sin el) / (TGD bulan – R * Cos el)

Terakhir yakni menghitung FIB dengan rumus:

FIB = (1 + cos i) /2

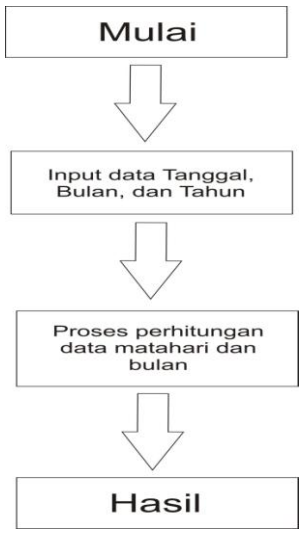
[22] *Ibid.* hlm. 360-361.

d. Diagram Alur Perhitungan Data *Ephemeris* Matahari dan Bulan Algoritma Jean Meeus

Dalam skripsi ini penulis menyusun sebuah aplikasi tentang data matahari dan bulan yang bernama *Zephemeris.* Aplikasi *Zephemeris* dirancang dengan menggunakan bahasa pemrograman *android*. Bahasa pemograman android merupakan pengembangan bahasa pemrograman *Java Micro 2 Edition (J2ME).* Pada android terdapat elemen-elemen yang telah disempurnakan, contohnya ialah *Dalvik Virtual Manager* yang dirancang dan dikostumisasi guna memastikan bahwa beberapa *feature-feature* berjalan lebih efisien pada perangkat *mobile.*[23]

Desain antar muka atau *interface* aplikasi *Zephemeris* dibuat dengan konsep *user friendly.* Pengguna hanya perlu meng-*input* data tanggal, bulan, dan tahun yang diinginkan, kemudian aplikasi *Zephemeris* akan menghitung lalu menampilkan hasilnya. Adapun diagram alur bekerjanya aplikasi *Zephemeris* sebagai berikut:

[23] Nazaruddin safaat. *Android "Pemrograman Aplikasi Mobile Smartphone dan Tablet PC Berbasis Android",* Bandung: Informatika, 2012. hlm. 4.



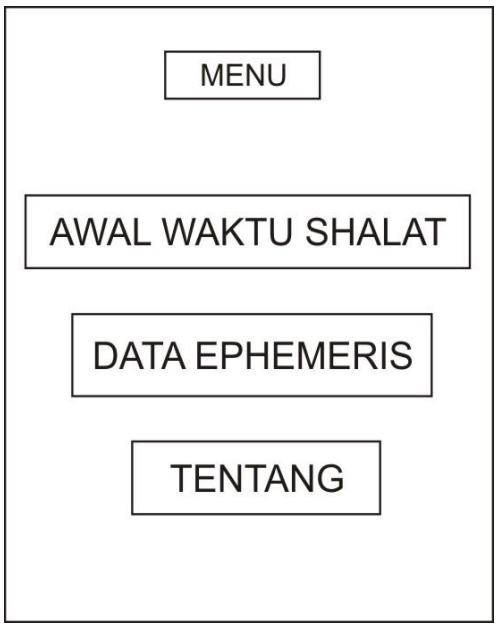
Gambar 3.1 *flowchart* umum dalam aplikasi *Zephemeris*

Dari gambar tersebut dapat dilihat alur pengoperasian aplikasi *Zephemeris* secara umum. Pertama, pengguna hanya tinggal menjalankan aplikasi lalu meng-*input* data tanggal, bulan dan tahun. Kedua, aplikasi secara otomatis akan melakukan proses perhitungan data matahari dan bulan pada tanggal yang telah di-*input*, dengan bahasa pemrograman *java*. Ketiga, data hasil perhitungan didalam aplikasi akan langsung ditampilkan di halaman selanjutnya.

B. Aplikasi *Zephemeris*

1. Desain Utama dan Spesifikasi Perangkat

a. Desain Antar Muka (*User Interface)*



Gambar 3.2 Tampilan Utama Menu Aplikasi

1) Awal Waktu Shalat

   Pada *activity* ini akan menampilkan kolom *input* yang berisi tanggal, bulan, tahun, *input manual*, kota, data lintang, data bujur, zona waktu dan tinggi tempat.

2) Data Ephemeris

   Pada *activity* ini akan menampilkan kolom *input* yang berisi tanggal, bulan, tahun, serta jam yang akan dihitung.

3) Tentang

   Pada halaman ini hanya akan menampilkan deskripsi singkat tentang aplikasi dan *developer*.

b. Spesifikasi Perangkat Keras

Dalam perancangan yang telah dijelaskan sebelumnya dibutuhkan beberapa hal seperti perangkat keras yakni untuk penyajian aplikasi. Adapun alat-alat yang dibutuhkan adalah:

1) *Mobile phone* Dengan Sistem Operasi Android

   Mobilephone digunakan untuk menjalankan program aplikasi yang telah dikembangkan. Adapun *mobile phone* yang digunakan adalah Asus Zenfone 4C dengan OS Android *Lollipop* (Android 5.0) sebagai contoh dalam menjalankan Aplikasi *Zephemeris*.

2) Laptop Byon Alverstone

   Laptop Byon Alverstone digunakan untuk merancang aplikasi *zephemeris* dengan spesifikasi chipset Intel(R) Core(TM) i3 2.27 Ghz, 2048MB RAM.

c. Spesifikasi Perangkat Lunak

Adapun perangkat lunak yang digunakan dalam proses pembuatan aplikasi perhitungan data *ephemeris* matahari dan bulan adalah sebagai berikut :

1) *Java Development Kit* (JDK) dan *Java Runtime Environment* (JRE) untuk instalisasi bahasa pemrograman Java.
2) *Integrated Development Environment* (IDE) *Eclipse Juno* yang berfungsi untuk mengolah bahasa pemrograman serta perhitungan yang dibutuhkan, sehingga memudahkan dalam pengembangan aplikasi.

3) *Android Software Development Kit (*Android SDK) yang berfungsi untuk menyediakan emulator dan perlengkapan lain dalam pembuatan sebuah aplikasi.
4) *Android Development Tools* (ADT) untuk memberikan kemudahan dalam pengembangan aplikasi, membuat tampilan aplikasi yang menarik, menambahkan komponen dan menyiapkan aplikasi yang telah dikembangkan untuk di distribusikan.
5) *Microsoft Windows 7* sebagai sistem operasi yang digunakan untuk merancang *layout* maupun *background* aplikasi.
6) *Corel Draw X4* untuk membuat tampilan yang menarik.
7) *Nox App Player* untuk menjalankan dan sebagai emulator aplikasi.

2. Implementasi Aplikasi *Zephemeris*

a. Implementasi Perhitungan Aplikasi

Setelah perancangan desain aplikasi *Zephemeris* sebagaimana telah dijelaskan diatas, tahap selanjutnya yakni pengimplementasian desain rancangan program tersebut ke dalam bahasa pemrograman *java* dengan menggunakan *Eclipse Juno 4.0*. Adapun *coding* implementasi perhitungan dalam aplikasi *Zephemeris* adalah sebagai berikut :

1) Implementasi Perhitungan *True Geocentric Distance*

```
//True Geocentric Distance
```

```
Double tgd =
((aa9+aa10*(t/10)+aa11*(t/10)*(t/10)+aa12*(t/10)*(t/10)*(t/10)+a
a13*(t/10)*(t/10)*(t/10)*(t/10))/100000000);

Double tgd1 = tgd * 100000000; Integer tgd2 = tgd1.intValue();
Double tgdm = tgd2 / 1000000000.0;

EL9 = ((-20.4898/3600.0)/tgd);

EL10 = (EL8+EL9+nut65)%360.0; //Bujur Matahari

Double sd = ((959.63/3600)/tgd);//Semi diamter
```

2) Implementasi Perhitungan *Equation of Time*

```
//equation

        aa6 = (((280.4664567+360007.6982779*(t/10)-
0.03032028*(t/10)*(t/10)+(t/10)*(t/10)*(t/10)/49931-
(t/10)*(t/10)*(t/10)*(t/10)/15299-
(t/10)*(t/10)*(t/10)*(t/10)*(t/10)/1988000)%360.0)*Math.PI/18
0)%360.0;

        equ = (((aa6*180/Math.PI)-0.0057183 - asen2 +
nut65*Math.cos(eps41*Math.PI/180))*4)%20.0; if (equ<-
16){epu= equ +20;}else {epu = equ;}
```

3) Implementasi Perhitungan *Apparent Declination*

```
// deklinasi

            dekl =
Math.asin(Math.sin((aa14/3600.0)*Math.PI/180)*Math.cos(ep
s41*Math.PI/180)+Math.cos((aa14/3600.0)*Math.PI/180)*Mat
h.sin(eps41*Math.PI/180)*Math.sin(EL10*Math.PI/180))*180/
Math.PI;

String dkm; if (dekl <0){dkm ="-";}else{dkm =" ";}
```

4) Implementasi Perhitungan *Right Ascension*

```
//asensiorekta

asen1 =
(Math.atan2(Math.sin(EL10*Math.PI/180)*Math.cos(eps41*Mat
h.PI/180)-
```

```
Math.tan((aa14/3600.0)*Math.PI/180)*Math.sin((aa14/3600.0)*
Math.PI/180),Math.cos(EL10*Math.PI/180)))*180/Math.PI;

if (asen1 < 0){asen2 = (asen1+360.0);}else {asen2 =
(asen1%360.0);}
```

5) Implementasi *Apparent Latitude*

```
//apparent lattitude

Double koki1 = (L1 - 2235*Math.sin(aa1) +
382*Math.sin(aa9) + 175*Math.sin(aa7 - aa5) +
175*Math.sin(aa7 + aa5) + 127*Math.sin(aa1 - aa4) -
115*Math.sin(aa1 + aa4))/1000000.0;

String alb; if (koki1 <0){alb ="-";}else{alb =" ";}
```

b. Implementasi Antarmuka Aplikasi

Bagian ini merupakan bagian uji hasil implementasi desain antarmuka setelah semua bahasa pemrograman ditulis dan dijalankan pada emulator android sehingga secara otomatis program Eclipse meng-*compile* aplikasi menjadi file *.apk* yang dapat diaplikasikan langsung pada *smartphone*. Karena aplikasi *Zephemeris* merupakan aplikasi perhitungan, maka tidak dibutuhkan koneksi internet untuk mengaksesnya. Aplikasi ini berbasis *offline*, pengguna hanya harus menginput data yang akan dihitung dan semua aktivitas perhitungan akan dijalankan didalam aplikasi.

Adapun implementasi dari aplikasi *Zephemeris* antara lain sebagai berikut :

Gambar 3.3 Desain Antarmuka Pada Tampilan Utama dan Halaman *Input* Untuk Perhitungan Data Ephemeris

Gambar 3.4 Desain Halaman *Input* Awal Waktu Shalat dan Tentang

Gambar 3.3 di atas merupakan hasil antarmuka dari tampilan *spalshscreen* pada aplikasi *Zephemeris* saat aplikasi baru dijalankan. Disebelahnya merupakan antarmuka tampilan yang menunjukan halaman untuk menginput data tanggal, bulan tahun, serta jam yang diinginkan untuk dihitung.

Selanjutnya, gambar 3.4 terdapat tampilan antarmuka yang akan ditemukan apabila memilih *list* menu "Awal Waktu Shalat" dan "Tentang". Pada halaman *input* waktu shalat pengguna hanya diharuskan memasukkan data sesuai yang diminta. Kemudian pada halaman "Tentang" hanya berisi deskripsi singkat tentang aplikasi dan *developer*.