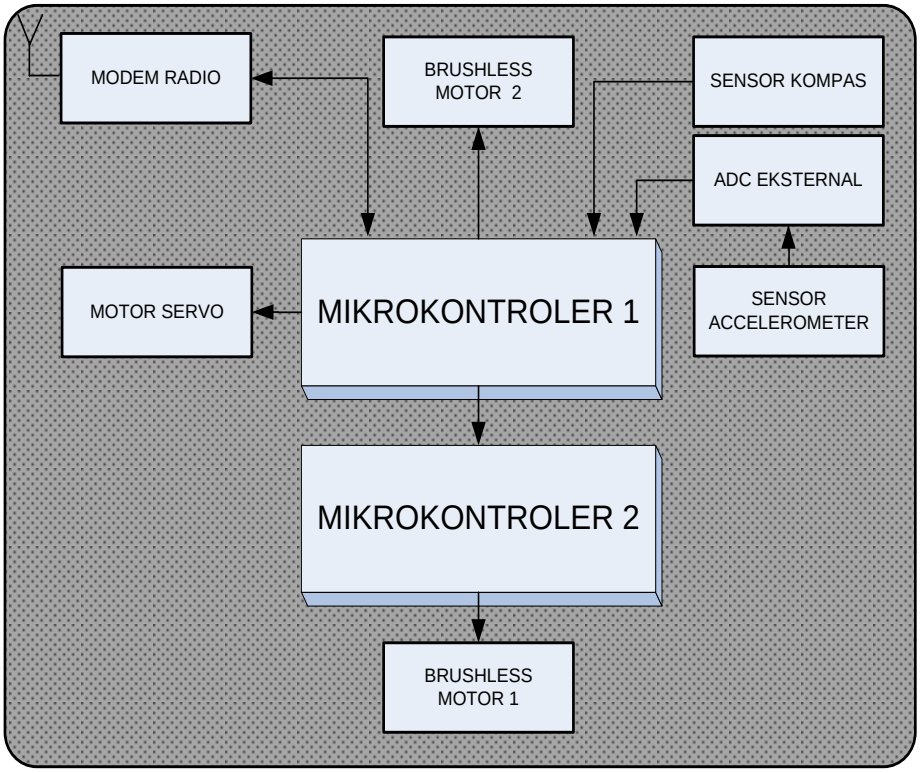
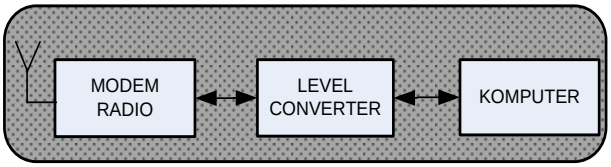
BAB III
PERANCANGAN SISTEM

3.1 Perancangan Secara Umum



(a)



(b)

Gambar 3.1. (a) Diagram blok sistem prototipe satelit, (b) Diagram blok sistem *interface* prototipe satelit

Berikut penjelasan blok-blok yang terdapat pada gambar di atas.

1. Mikrokontroler 1

   Blok ini merupakan mikrokontroler *basic stamp* BS2P40 yang berfungsi mengolah data dari yang diterima dari sensor kompas dan *accelerometer*.

Blok ini mengatur komunikasi antara mikrokontroler 2 dan komunikasi dengan PC.

2. Mikrokontroler 2

   Blok ini memuat mikrokontroler *basic stamp* BS2P40 yang berfungsi merima data sensor kompas dan mengolahnya untuk mengontrol *brushless* motor besar sehingga dapat mengejar *setpoint* yang diinginkan.

3. Sensor Kompas

   Blok ini memuat sensor hitachi HM55B yang berfungsi mengetahui arah gerak prototipe satelit dan datanya dikirimkan mikrokontroler 2 agar dapat diolah.

4. Sensor *Accelerometer*

   Blok ini memuat sensor MMA3201EG yang berfungsi untuk mengetahui kemiringan yang dialami oleh prototipe satelit dan datanya diubah oleh ADC eksternal lalu diolah oleh mikrokontroler 1.

5. ADC Eksternal

   Blok ini memuat IC ADC0832 yang berfungsi mengubah data analog dari sensor *accelerometer* manjadi data digital.

6. Modem Radio

   Blok ini memuat YS-1020UB yang berfungsi untuk mengirimkan data sensor yang diolah mikrokontroler 1 melaui sinyal radio.

7. *Brushless* motor 1

   Blok ini memuat *brushless* motor besar yang berfungsi sebagai penggerak dari prototipe satelit agar dapat mengejar dan mempertahankan *setpoint*, dengan data kompas yang diterima dari mikrokontroler 1.

8. *Brushless* motor 2

   Blok ini memuat *brushless* motor kecil yang berfungsi sebagai penggerak agar kesetimbangan pra prtotipe satelit tetap pada posisinya.

9. Motor *Servo*

   Blok ini memuat motor *servo* hitec HS-5245MG yang berfungsi menggerakkan dan mempertahankan sayap pada prototipe agar saat mengejar *setpoint* dapat berjalan.

10. Komputer

    Blok ini berfungsi menampilkan *interface* agar pengiriman dan penerimaan data dapat dilakukan dengan baik dan menyimpan data yang diterima dari prototipe satelit.

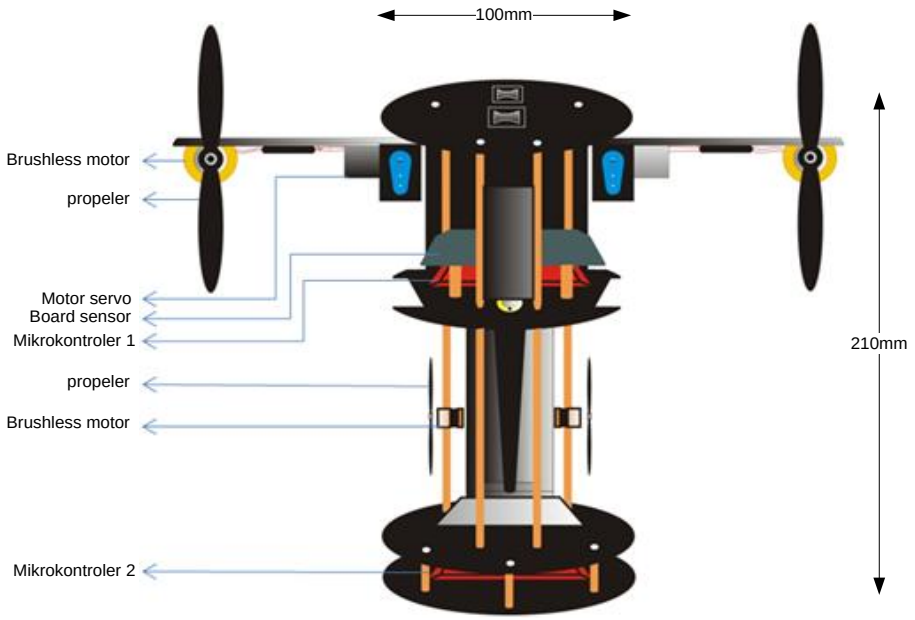
Pada perancangan prototipe satelit terjadi 2 input dari sensor kompas dan *accelerometer* dengan output untuk mengejar *setpoint* dan kesetimbangan. Sehingga sistem rancangan kontrol prototipe satelit menggunakan *multiple input multiple output* (MIMO).

3.2 Perancangan Perangkat Keras (Hardware)

3.2.1 Perancangan Mekanik

Bentuk dari prototipe satelit menyerupai tabung dengan ukuran yaitu sebagai berikut :

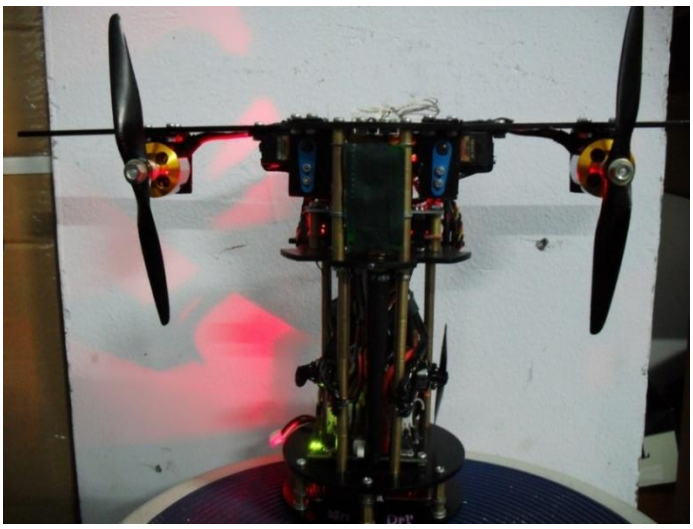
1. Diameter = 100mm
2. Panjang = 210mm
3. Berat = 1200 g



Gambar 3.2. Bentuk prototipe satelit dalam 3D

Namun bentuk disesuaikan dengan roket yang akan diterbangkan. Sementara bentuk prototipe satelit mengadaptasi dari *payload* yang dilombakan pada kontes roket Indonesia. Panjang tali dari prototipe satelit ketiang gantungan 204 mm. Perancangan prototipe satelit menggunakan bahan akrilik sebagai pemberi sekat antara komponen yang digunakan dan sayap atau tangan, serta *spacer* sebagai penyangga dari prototipe satelit. Peletakan komponen sangat penting sehingga tidak mengganggu jalannya prototipe satelit. Peletakan komponen sebagai berikut :

1. Bagian atas digunakan untuk peletakan *basic stamp* 1, modem radio, ESC untuk motor besar, sensor kompas dan sensor *accelerometer* .
2. Bagian bawah digunakan untuk peletakan *basic stamp* 2 dan sistem sumber tegangan yang diterima dari baterai *lithium polymer*. *Brushless* motor kecil dan ESCnya sebanyak 2 buah di samping.
3. Motor *servo*, *Brushless* motor dengan jumlah 2 buah masing-masing diletakkan pada bagian sayap atau tangan.



Gambar 3.3. Bentuk prototipe satelit

3.2.2 Mikrokontroler Basic Stamp (BS2P40)

Mikrokontroler yang digunakan adalah mikrokontroler jenis *basic stamp* BS2P40 dengan dilengkapi 40 *port* yang dapat digunakan sebagi *input* ataupun

output. Pemilihan mikrokontroler jenis ini didasari kemampuannya yang cukup handal, permograman yang tidak terlalu sulit dan harganya yang relatif tidak terlalu mahal. Berikut konfigurasi *port*-nya yang dipakai.

Tabel 3.1. Konfigurasi port pada mikrokontroler *basic stamp* 1

Nama Pin	No. Pin	Fungsi	Keterangan
MAIN I/O	0	*in&out*	di/do (sensor kompas)
MAIN I/O	1	*output*	Tx (komunikasi antar mikrokontroler)
MAIN I/O	2	*output*	*clock* (sensor kompas)
MAIN I/O	3	-	-
MAIN I/O	4	*output*	*enable* (sensor kompas)
MAIN I/O	5	*input*	RX(komunikasi antar radio frekuensi)
MAIN I/O	6	*output*	cs (ADC)
MAIN I/O	7	*output*	Tx(komunikasi antar radio frekuensi)
MAIN I/O	8	*input*	di (ADC)
MAIN I/O	9	*output*	*brushless* motor kecil kiri
MAIN I/O	10	*output*	do (ADC)
MAIN I/O	11	*output*	*brushless* motor kecil kanan
MAIN I/O	12	*output*	*clock* (ADC)
MAIN I/O	13	*output*	motor *servo* kiri
MAIN I/O	14	-	-
MAIN I/O	15	*output*	motor *servo* kanan

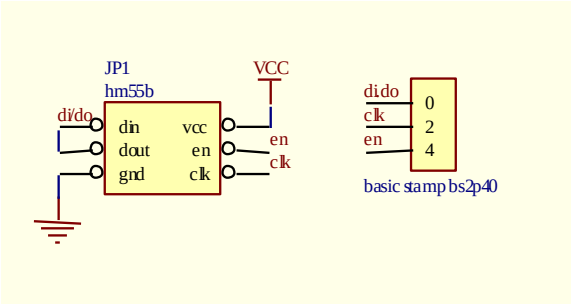
Tabel 3.2. Konfigurasi port pada mikrokontroler *basic stamp* 2

Nama Pin	No. Pin	Fungsi	Keterangan
MAIN I/O	0	*output*	*brushless* motor besar kanan
MAIN I/O	1	*input*	Rx (komunikasi antar mikrokontroler)
MAIN I/O	2	*output*	*brushless* motor besar kiri
MAIN I/O	3 s/d 15	-	-

3.2.3 Sensor Kompas HM55B

Sensor kompas yang digunakan dalam prototipe satelit adalah jenis hitachi HM55B. Pemilihan sensor ini didasari oleh kemampuannya yang cukup handal dan kemampuan dalam pembacaan data yang cepat antara 30-40 ms dari pembacaan awal sampai data siap untuk dibaca kembali. Pembacaan yang cepat tersebut sangat dibutuhkan dalam pencarian arah gerak prototipe satelit.

Penggunaan modul hitachi HM55B juga sangat sederhana karena tidak memerlukan rangkaian tambahan dalam pemakaiannya. Berikut rangkaian hitachi HM55B dengan mikrokontroler *basic stamp* BS2P40 yang diperlihatkan pada gambar 3.4



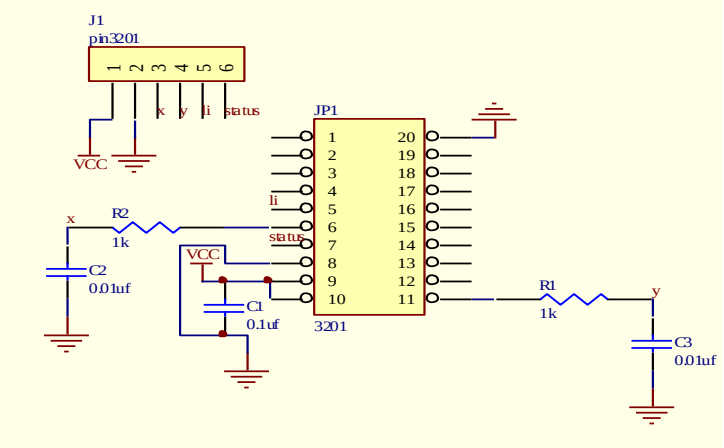
Gambar 3.4. Rangkaian pin sensor kompas hitachi HM55B dihubungkan dengan BS2P40

Pada modul hitachi HM55B akan mengirimkan data axis x dan axis y, kedua data axis ini akan diolah oleh mikrokontroler *basic stamp* 1 yang kemudian diubah dalam satuan derajat dengan menggunakan persamaan di bawah ini.

$$Sudut\ \theta = \arctan\left(-(\text{axis y})/(\text{axis x})\right) \quad \text{.....3.1}$$

3.2.4 Sensor Accelerometer MMA3201EG

Sensor *accelerometer* yang digunakan MMA3201 dengan menghasilkan data analog.

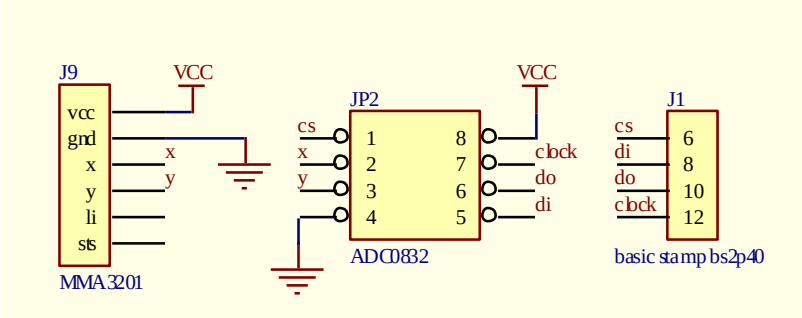


Gambar 3.5. Rangkaian MMA3201

Sensor ini berfungsi untuk mendeteksi kemiringan pada prototipe satelit sehingga kembali ke posisi yang sempurna. Karena dengan gangguan seperti angin prototipe satelit akan berada pada posisi tidak stabil atau sempurna sehingga menggunakan sensor ini, sehingga jika terjadi kemiringan yang tidak diinginkan maka motor yang berada di samping akan melakukan perputaran agar ke posisi yang stabil atau sempurna. Karena data yang di keluarkan adalah analog maka di perlukan ADC untuk mengubah datanya menjadi digital.

3.2.5 ADC (Analog to Digital Converter)

ADC yang dipilih untuk perancangan prototipe satelit adalah IC ADC0832. Pemilihan ADC jenis ini dikarenakan kebutuhan pada prototipe satelit yang memiliki 2 *channel* yang dipergunakan untuk sensor *accelerometer* yaitu axis x dan axis y. ADC0832 mempunyai tegangan Vref yang terhubung langsung dengan Vsumber, sehingga mudah dalam penggunaannya tanpa rangkaian tambahan. Berikut rangkaian ADC0832 yang dihubungkan langsung dengan *basic stamp* BS2P40 yang diperlihatkan pada gambar 3.6



Gambar 3.6. Rangkaian ADC0832

Bagian *multiplexer* ini mempunyai 2 buah masukan, setiap masukan memilki alamat sendiri sehingga dapat dipilih secara terpisah melalui *address* A0 dan A1. Tabel 3.3 menunjukkan alamat *multiplexer* dari masing-masing masukan *channel***.**

Tabel 3.3. *Mux addressing*: ADC0832 *single-ended mux mode*

MUX address		Channel	
SGL/DIF	**ODD/SIGN**	**0**	**1**
1	0	+	
1	1		+

3.2.6 ESC (Electronic Speed Controller)

ESC (*Electronic Speed Controller*) merupakan sebuah modul rangkaian elektronik yang fungsinya mengatur putaran pada motor sesuai *ampere* yang dibutuhkan oleh motor. ESC dipengaruhi dan bekerja untuk untuk 2 beban yaitu bobot dan motor.

Ampere ESC harus lebih besar atau minimal sama dengan *ampere brushless* motor dan *ampere* ESC diusahakan menyesuaikan bobot atau minimal sesuai dengan yang dibutuhkan. Jumlah ESC sesuai dengan jumlah motor yang dipakai yaitu sebanyak 4 buah ESC.

3.2.7 Brushless motor

Motor yang digunakan adalah *brushless* motor yang berfungsi sebagai penggerak untuk propeler atau baling-baling. *Brushless* motor berbeda dengan motor DC *brush* yang masih menggunakan sikat pada bagian dalam motor dan dapat menimbulkan *electrical noise*. Sedangkan dengan menggunakan *brushless* motor tidak menimbulkan *electrical noise* dan dapat bertahan lama dibandingkan dengan motor *brush*.

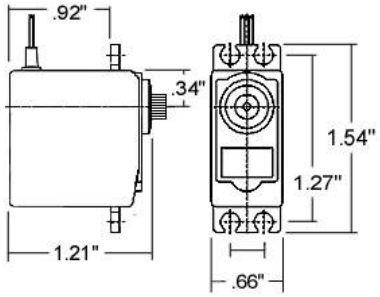
Kecepatan putaran *brushless* motor yang keluar dari ESC diatur oleh pulsa dari mikrokontroler. Jika tidak diberi pulsa maka tidak akan bekerja walaupun tegangan sudah diberikan. Dengan putaran dari *brushless* motor maka propeler atau baling-baling akan berputar sehingga dapat menggerakkan prototipe satelit.

Jumlah dari *brushless* motor dan propeler yang dipakai adalah 4 buah. Dua buah pada bagian sayap atau tangan yang berfungsi untuk mengarahkan ke kiri atau kanan, dua buah di bawah pada bagian samping sebagai penyetimbang agar berada pada posisi tetap.

3.2.8 Motor Servo

Motor *servo* berfungsi untuk menggerakkan bagian sayap/tangan yang akan bergerak 90º. Pemilihan motor *servo* HS-5245MG didasarkan dengan *torque* mulai 4,4 kg sampai 5,5 kg sesuai dengan tegangan yang diberikan antara 4,8 volt sampai 6 volt.

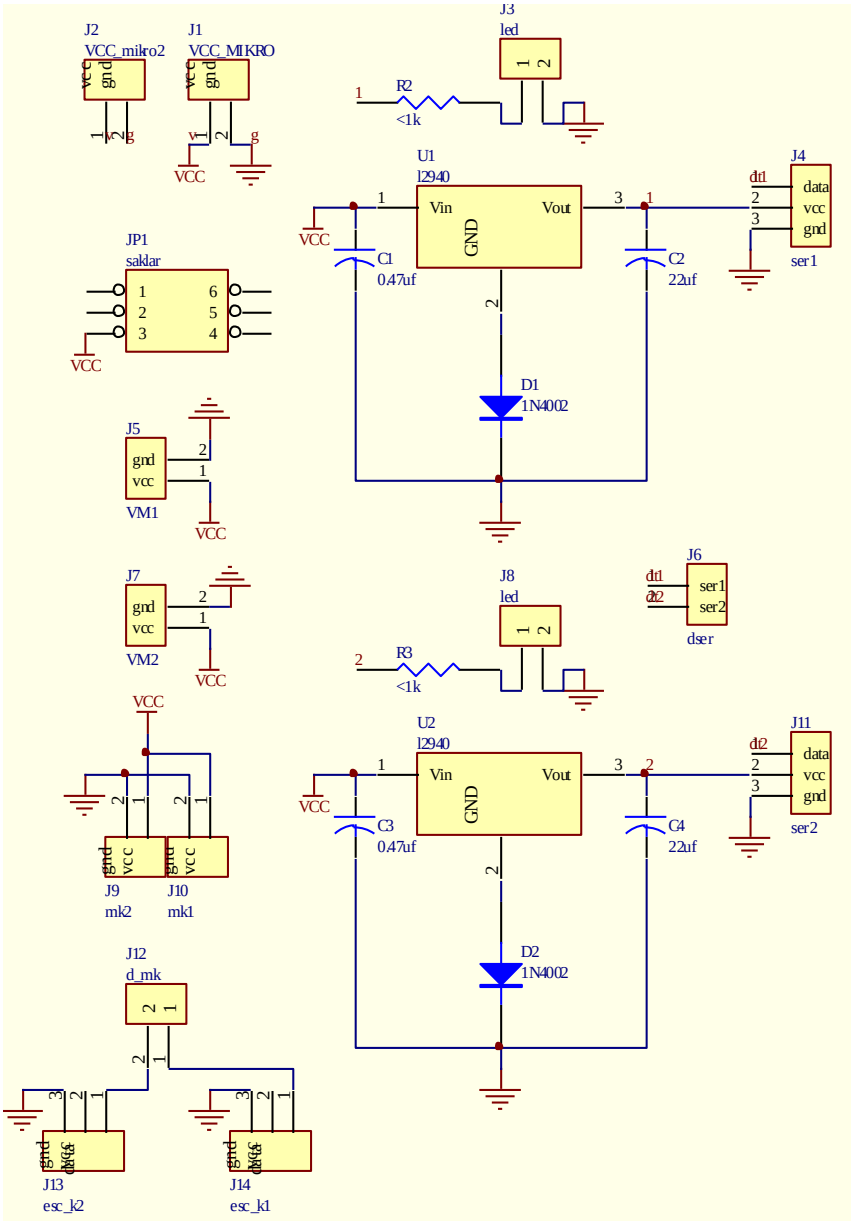
Motor *servo* akan bekerja dengan memberikan tegangan dan pulsa sesuai yang diinginkan. Setelah tegangan dan pulsa diberikan motor *servo* akan menahan dengan spesifikasi *torque*-nya. Jumlah motor *servo* yang dipakai adalah 2 buah yang dipasang pada masing-masing bagian sayap atau tangan.



Gambar 3.7. Bentuk dan ukuran motor *servo* HS-5245MG

3.2.9 Sumber Tegangan

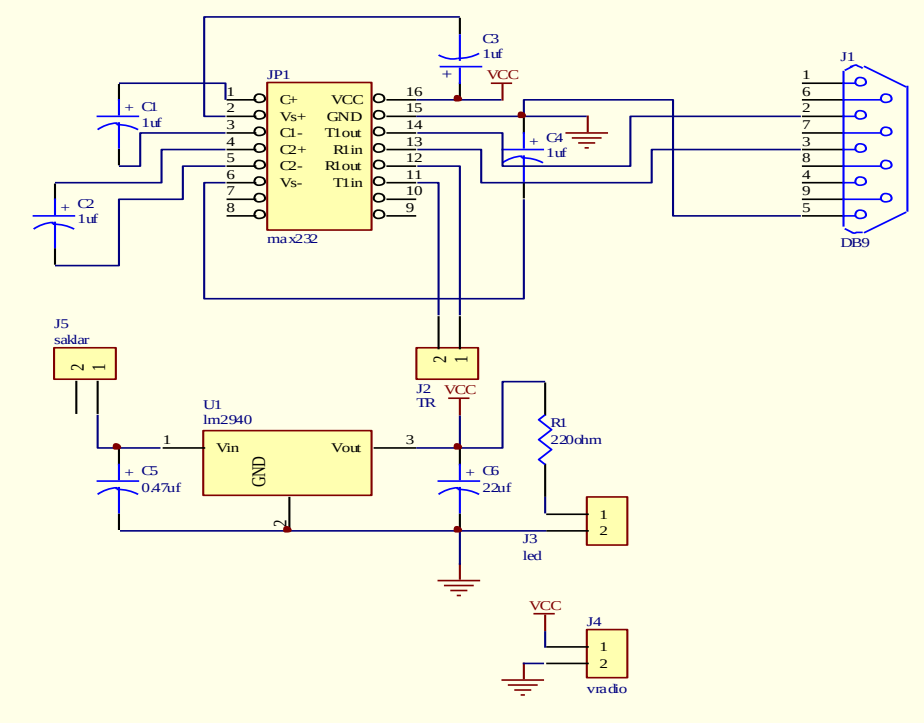
Lithium polymer adalah baterai yang menjadi sumber tegangan untuk komponen-komponen pada prototipe satelit. Baterai *lithium polymer* yang digunakan adalah 11,1 volt dengan 2,2 *ampere*. Beberapa komponen memerlukan sumber tegangan dari regulator agar dapat bekerja. Penggunaan baterai *lithium polymer* digunakan juga untuk *brushless* motor yang kecil dan *brushless* motor besar. IC yang digunakan adalah LM2940CT yang berfungsi menurunkan tegangan menjadi ±5 Volt agar stabil.



Gambar 3.8. Rangkaian sumber tegangan untuk prototipe satelit

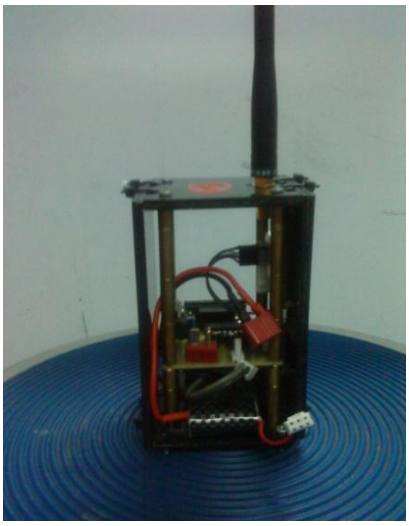
3.2.10 Modul Komunikasi

Konektor pada komputer yaitu DB-9 akan mengeluarkan data dengan level tegangan RS232, sehingga membutuhkan IC jika komunikasi menggunakan tegangan TTL. IC Max232 berfungsi untuk mengubah tegangan dari TTL menjadi level RS232. Sehingga komputer dapat berkomunikasi dengan prototipe satelit. Rangkaian IC Max232 dapat dilihat pada gambar 3.9:



Gambar 3.9. Rangkaian ICMAx232

IC yang dipakai pada sistem ini memiliki 16 pin. Agar dapat dihubungkan dengan *port* serial PC dan pada terminal TTL, maka IC ini memerlukan komponen tambahan berupa kapasitor. Rangkaian di atas beroperasi dengan tegangan 5 volt.



Gambar 3.10. *Ground Segment*

3.2.11 Radio

Modem radio yang digunakan untuk berkomunikasi antara komputer dengan prototipe satelit adalah YS-1020UB. Modem ini berkomunikasi melalui frekuensi radio YS-1020UB yang mempunyai 8 kanal dengan frekuensi yang berbeda dan jarak jangkauannya hingga 800 meter. Berikut ini adalah tabel yang menjelaskan konfigurasi pin-pin pada radio YS-1020UB.

Tabel 3.4. Pin-Pin modul radio YS-1020UB

Pin	Nama Pin	Fungsi	*Level*
1	GND	*Ground*	
2	Vcc	Tegangan *Input*	+3.3 s/d 5.5 V
3	RXD/TTL	*Input* Serial Data	TTL
4	TXD/TTL	*Output* Serial Data	TTL
5	DGND	*Digital Grounding*	
6	A(TXD)	Aof RS-485 or TXD of RS-232	A (RXD)
7	B(RXD)	B of RS-485, RXD or RS-232	B (TXD)
8	SLEEP	*Sleep Control (Input)*	TTL
9	Test	*Testing*	

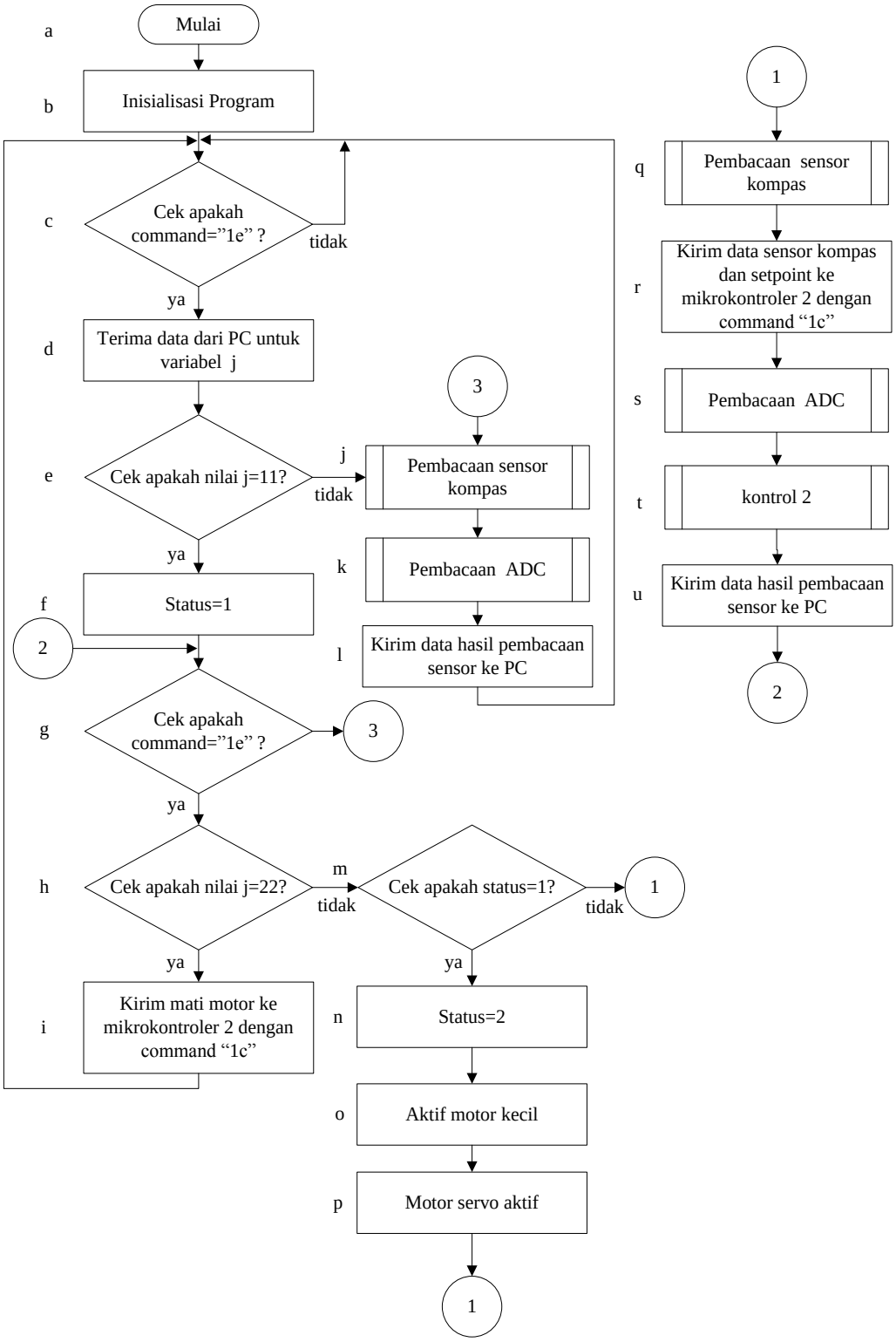
Dari konfigurasi pin-pin radio YS-1020UB, pin yang dipakai pada sistem ini hanya terdiri dari GND, Vcc, RXD/TTL dan TXD/TTL. Penggunaan radio ini sangat penting karena pengiriman dan penerimaan data dilakukan melalui udara dengan jarak yang jauh, Sehingga dapat dilakukan tanpa adanya penggunaan kabel yang tersambung antara *ground segment* dan prototipe satelit.

3.3 Perangkat Lunak (Software)

3.3.1 Perancangan Algoritma Keseluruhan

Perancangan algoritma sangat penting untuk setiap sistem kendali. Sehingga dibuat algoritma keseluruhan untuk sistem kendali prototipe satelit agar dapat berjalan sesuai dengan keinginan. Berikut adalah algoritma dalam bentuk diagram alir:

1. Algoritma mikrokontroler satu

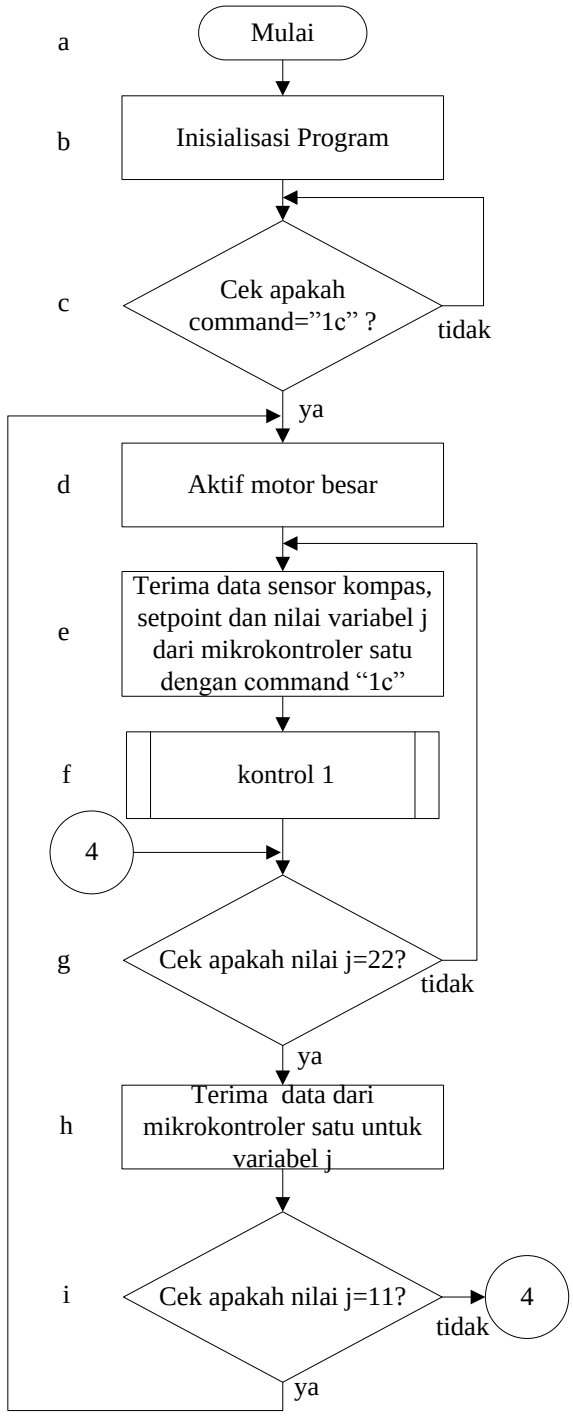


Gambar 3.11. Diagram alir mikrokontroler satu

Tabel 3.5. Penjelasan diagram alir mikrokontroler satu

Urutan	Keterangan
a	Memulai awal program.
b	Pendeklarasian program. Pengaturan nama prosesor, nama *compiler*, konfigurasi pin-pin dan deklarasi variabel.
c	Mengecek apakah ada *command* "1e" yang diterima. Jika ya akan melanjutkan ke urutan d, jika tidak akan kembali ke urutan c sampai menerima *command* "1e".
d	Menerima data dari PC yang disimpan pada variabel j
e	Mengecek apakah nilai pada variabel j=11. Jika ya akan melanjutkan ke urutan f, jika tidak akan ke urutan j
f	memberikan nilai 1 pada variabel status.
g	Mengecek apakah ada *command* "1e" yang diterima. Jika ya akan melanjutkan ke urutan h, jika tidak akan ke urutan j.
h	Mengecek apakah nilai pada variabel j=22. Jika ya akan melanjutkan ke urutan i, jika tidak akan ke urutan m
i	Mengirimkan data mati motor besar ke mikrokontroler dua dengan command "1c".
j	Memanggil prosedur pembacaan sensor kompas
k	Memanggil prosedur pembacaan ADC (sensor *accelerometer*)
l	Kirim data sensor kompas dan *accelerometer* ke PC melalui radio frekuensi dan dikirim dengan *command* "1d"
m	Mengecek apakah nilai variabel status = 1. Jika ya akan melanjutkan ke urutan n, jika tidak akan ke urutan q.
n	Memberikan nilai variabel status=2.
o	Mengaktifkan motor yang kecil
p	Motor *servo* aktif dan membuka sayap kanan dan kiri pada prototipe satelit
q	Memanggil prosedur pembacaan sensor kompas
r	Mengirimkan data pembacaan sensor kompas dan data *setpoint* ke mikrokontroler dua.
s	Memanggil prosedur pembacaan ADC untuk sensor *accelerometer*.
t	Memenggil prosedur pembacaan kontrol dua untuk motor kecil.
u	Kirim data sensor kompas dan *accelerometer* ke PC melalui radio frekuensi dan kirim dengan *command* "1d"

2. Algoritma mikrokontroler dua



Gambar 3.12. Diagram alir mikrokontroler dua

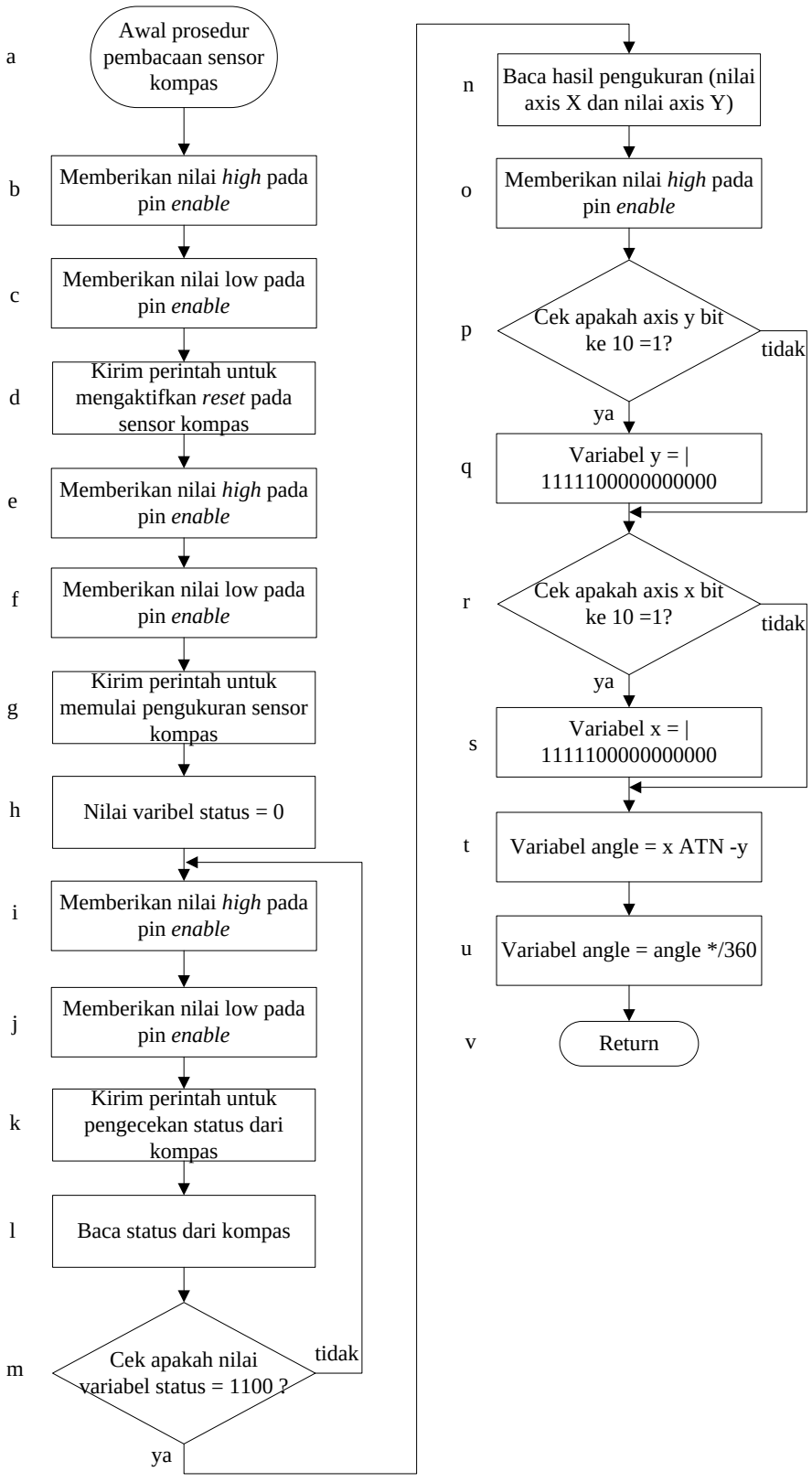
Tabel 3.6. Penjelasan diagram alir mikrokontroler dua

Urutan	Keterangan
a	Memulai awal program.
b	Pendeklarasian program. Pengaturan nama prosesor, nama *compiler*, konfigurasi pin-pin dan deklarasi variabel.
c	Mengecek apakah ada *command* "1c" yang diterima dari mikrokontroler satu . Jika ya akan melanjutkan ke urutan d, jika tidak akan kembali ke urutan c sampai menerima *command* "1c".
d	Mengaktifkan motor besar
e	Menerima data sensor kompas, *setpoint* dan nilai untuk variabel j dari mikrokontrol satu.
f	Memanggil prosesur pembacaan kontrol 1
g	Mengecek apakah nilai pada variabel j=22. Jika ya akan melanjutkan ke urutan i, jika tidak akan ke urutan f
h	Menerima data dari mikrokontroler satu untuk variabel j
i	Mengecek apakah nilai pada variabel j=11. Jika ya akan melanjutkan ke urutan d, jika tidak akan ke urutan g.

3.3.2 Perancangan Prosedur

Dalam perancangan algoritma prototipe satelit terdapat prosedur agar tidak terjadi pengulangan yang terlalu banyak dalam program. Berikut adalah perancangan prosedur untuk prototipe satelit :

1. Prosedur pembacaan sensor kompas

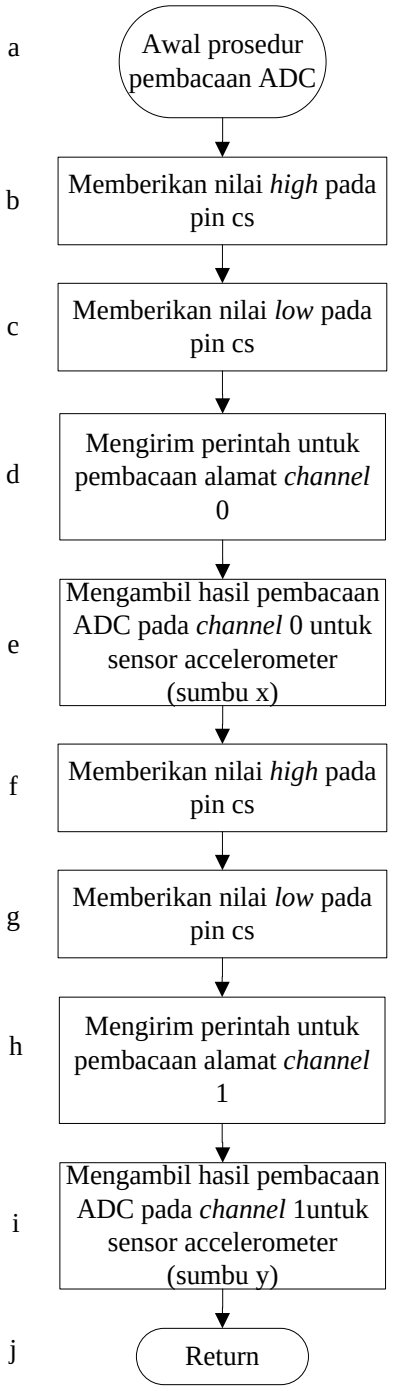


Gambar 3.13. Diagram alir prosedur sensor kompas

Tabel 3.7. Penjelasan diagram alir prosedur sensor kompas

Urutan	Keterangan
a	Memulai prosedur diagram alir pembacaan sensor kompas
b	Memberikan nilai *high* pada pin *enable* sensor kompas.
c	Memberikan nilai *low* pada pin *enable* sensor kompas.
d	Kirim perintah untuk mengaktifkan *reset* pada sensor kompas.
e	Memberikan nilai *high* pada pin *enable* sensor kompas.
f	Memberikan nilai *low* pada pin *enable* sensor kompas.
g	Kirim perintah untuk memulai pengukuran sensor kompas
h	Isi variabel status dengan nilai 0.
i	Memberikan nilai *high* pada pin *enable* sensor kompas.
j	Memberikan nilai *low* pada pin *enable* sensor kompas.
k	Kirim perintah untuk pengecekan status dari sensor kompas.
l	Kirim perintah untuk membaca status sensor kompas.
m	Mengecek apakah nilai variabel status = 1100. Jika ya akan ke ururan n, jika tidak akan ke urutan i.
n	Kirim perintah untuk membaca hasil pengukuran axis X dan axis Y.
o	Memberikan nilai *high* pada pin *enable* sensor kompas.
p	Mengecek apakah nilai pada variabel y bit ke-10 = 1. Jika ya lanjutkan ke urutan q, jika tidak lanjutkan ke urutan r.
q	Nilai variabel y di-OR kan dengan nilai 1111100000000000, kemudian hasilnya disimpan kembali pada variabel y.
r	Mengecek apakah nilai pada variabel x bit ke-10 = 1, jika ya lanjutkan ke urutan s, jika tidak lanjutkan ke poin t.
s	Nilai variabel x di-OR kan dengan nilai 1111100000000000, kemudian hasilnya disimpan kembali pada variabel x.
t	Isi variabel *angle* dengan hasil dari x ATN -y.
u	Hasil dari ((*angle*)*/360) simpan pada variabel *angle*.
v	Keluar dari prosedur dan kembali mengeksekusi instruksi selanjutnya pada program utama pada mikrokontroler satu.

2. Prosedur pembacaan ADC

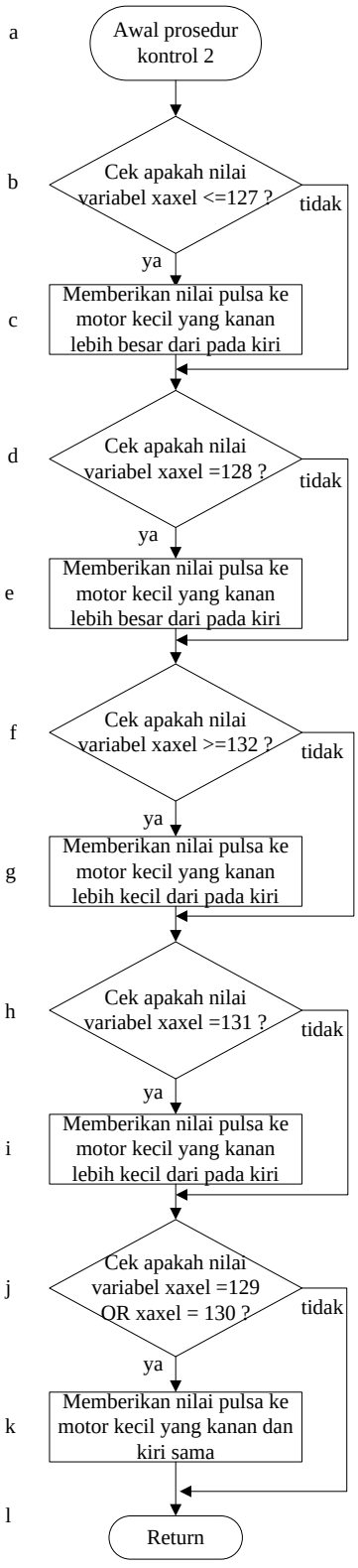


Gambar 3.14. Diagram alir prosedur pembacaan ADC

Tabel 3.8. Penjelasan diagram alir prosedur pembacaan ADC

Urutan	Keterangan
a	Memulai prosedur diagram alir pembacaan ADC
b	Memberikan nilai *high* pada pin cs sensor *accelerometer.*
c	Memberikan nilai *low* pada pin cs sensor *accelerometer*.
d	Mengirim perintah untuk pembacaan alamat *channel* 0
e	Mengambil hasil pembacaan ADC pada *channel* 0 untuk sensor *accelerometer* (sumbu x)
f	Memberikan nilai *high* pada pin cs sensor *accelerometer.*
g	Memberikan nilai *low* pada pin cs sensor *accelerometer*.
h	Mengirim perintah untuk pembacaan alamat *channel* 1
i	Mengambil hasil pembacaan ADC pada *channel* 1 untuk sensor *accelerometer* (sumbu y).
j	Keluar dari prosedur dan kembali mengeksekusi intruksi selanjutnya pada program utama pada mikrokontroler satu.

3. Prosedur pembacaan kontrol 2 (motor kecil)

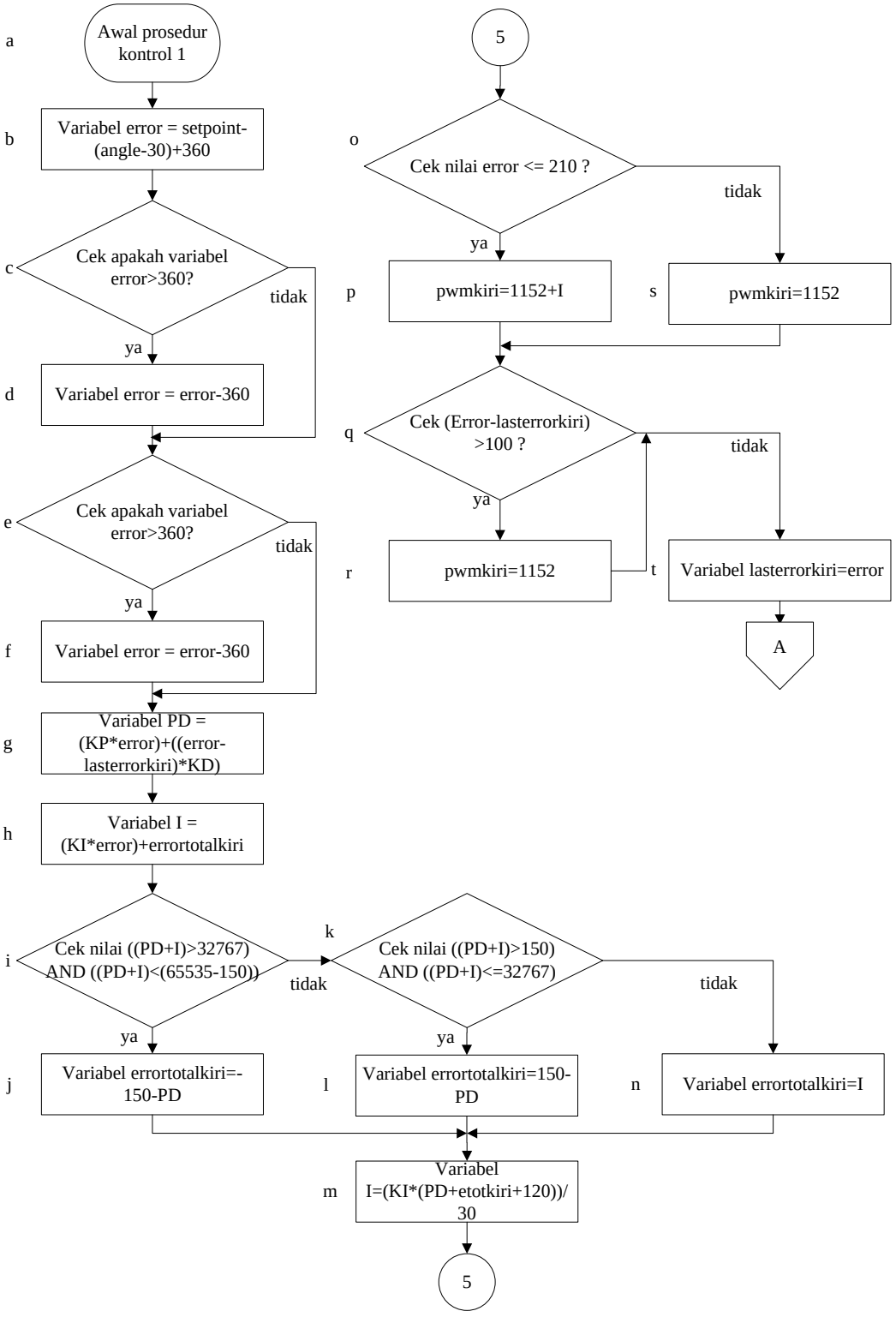


Gambar 3.15. Diagram alir prosedur kontol 2 (motor kecil)

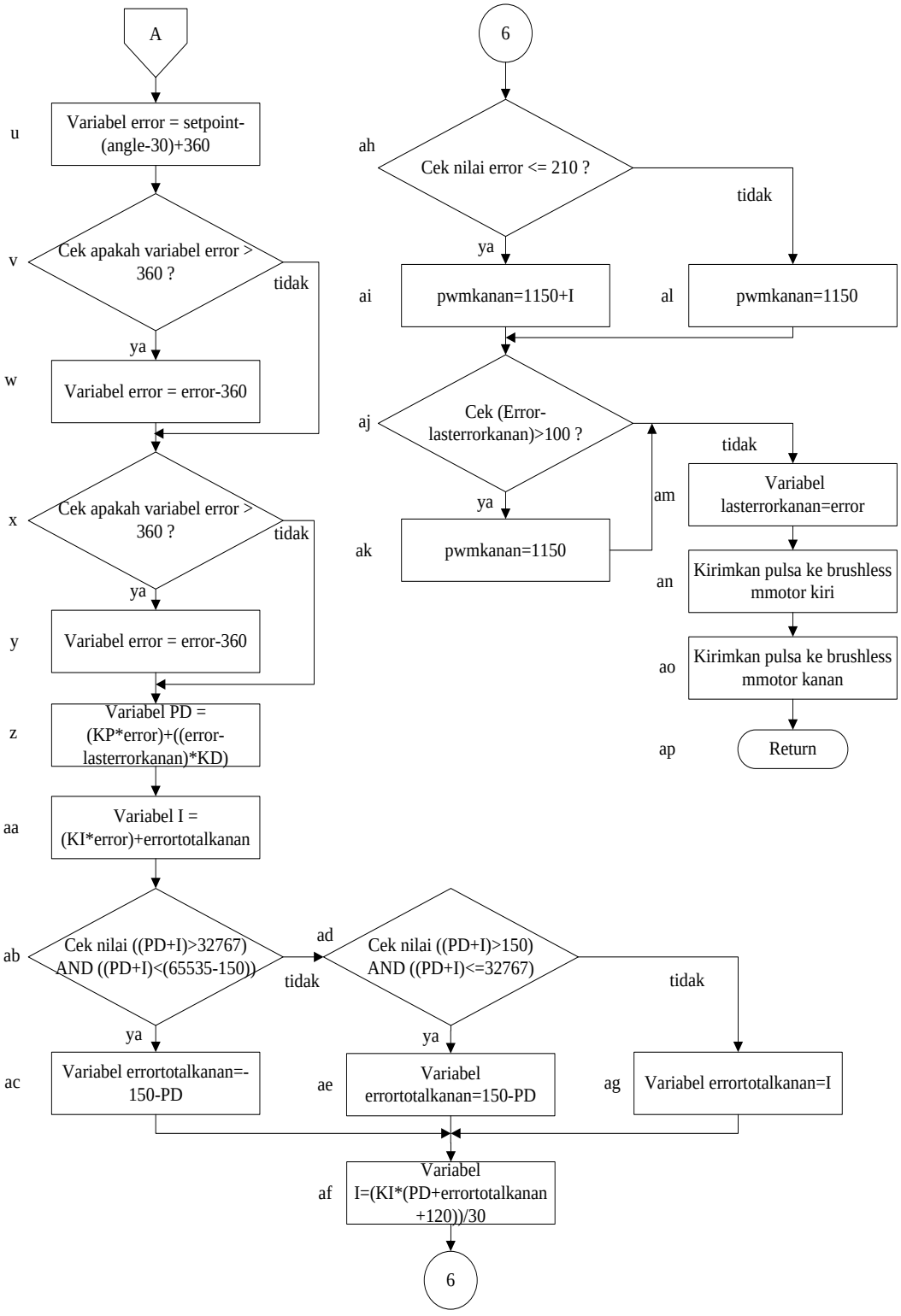
Tabel 3.9. Penjelasan diagram alir prosedur kontrol 2 (motor kecil)

Urutan	Keterangan
a	Memulai prosedur diagram alir pembacaan kontrol 2 untuk motor kecil
b	Mengecek apakah nilai variabel xaxel <= 127. Jika ya akan ke urutan c, jika tidak akan melanjutkan ke urutan d.
c	Memberikan nilai pulsa ke variabel motor kanan dan kiri dengan nilai motor kanan lebih besar dari pada nilai motor kiri.
d	Mengecek apakah nilai variabel xaxel = 128. Jika ya akan ke urutan e, jika tidak akan melanjutkan ke urutan f.
e	Memberikan nilai pulsa ke variabel motor kanan dan kiri dengan nilai motor kanan lebih besar dari pada nilai motor kiri.
f	Mengecek apakah nilai variabel xaxel >= 132. Jika ya akan ke urutan g, jika tidak akan melanjutkan ke urutan h
g	Memberikan nilai pulsa ke variabel motor kanan dan kiri dengan nilai motor kanan lebih kecil dari pada nilai motor kiri.
h	Mengecek apakah nilai variabel xaxel = 131. Jika ya akan ke urutan i, jika tidak akan melanjutkan ke urutan j.
i	Memberikan nilai pulsa ke variabel motor kanan dan kiri dengan nilai motor kanan lebih kecil dari pada nilai motor kiri.
j	Mengecek apakah nilai variabel xaxel >128 or < 132. Jika ya akan ke urutan k, jika tidak akan melanjutkan ke urutan l.
k	Memberikan nilai pulsa ke variabel motor kanan dan kiri dengan nilai motor kanan sama dengan nilai motor kiri.
l	Keluar dari prosedur dan kembali mengeksekusi instruksi selanjutnya pada program utama pada mikrokontroler satu.

4. Prosedur pembacaan kontrol 1 (motor besar)



Gambar 3.16. Diagram alir pembacaan kontrol 1 untuk motor besar



Gambar 3.17. Diagram alir pembacaan kontrol 1 untuk motor besar (lanjutan)

Tabel 3.10. Penjelasan diagram alir pembacaan kontrol 1 (motor besar)

Urutan	Keterangan
a	Memulai prosedur diagram alir pembacaan kontrol 2 untuk motor kecil
b	Hasil dari (*setpoint*-(*angle*-30)+360) simpan ke variabel *error*.
c	Mengecek apakah variabel *error* > 360. Jika ya akan melanjutkan ke urutan d, jika tidak akan melanjutkan ke urutan e.
d	Hasil dari (*error*-360) simpan kembali pada variabel *error*.
e	Mengecek apakah variabel *error* > 360. Jika ya akan melanjutkan ke urutan f, jika tidak akan melanjutkan ke urutan g.
f	Hasil dari (*error*-360) simpan kembali pada variabel *error*.
g	Hasil dari (KP**error*)+((*error*-lasterrorkiri)*KD) simpan ke Variabel PD.
h	Hasil dari ((KI**error*)+*error*totalkiri) simpan ke Variabel I.
i	Mengecek apakah nilai (PD+I) > 32767) dan (PD+I) < (65535-150). Jika ya akan melanjutkan ke urutan j, jika tidak akan melanjutkan ke urutan k.
j	Kurangi nilai (-150) dengan nilai pada variabel PD, lalu simpan hasilnya ke variabel *error*totalkiri. Setelah itu, melanjutkan ke urutan m.
k	Mengecek apakah nilai (PD+I) > 150 dan (PD+I) <= 32767. Jika ya akan melanjutkan ke urutan l, jika tidak akan melanjutkan ke urutan n.
l	Kurangi nilai (150) dengan nilai pada variabel PD, lalu simpan hasilnya ke variabel *error*totalkiri. Setelah itu, melanjutkan ke urutan m.
m	Hasil dari (KI*(PD+*error*totalkiri+120))/30) simpan ke variabel I. Setelah itu, lanjutkan ke urutan o.
n	Isi variabel *error*totalkiri dengan nilai pada variabel I. Setelah itu, melanjutkan ke urutan m.
o	Mengecek apakah nilai variabel *error* <= 210. Jika ya akan melanjutkan ke urutan p, jika tidak akan melanjutkan ke urutan s.
p	Tambal nilai pada variabel I dengan nilai 1152 lalu hasilnya simpan kedalam variabel pwmkiri.
q	Mengecek apakah nilai (*Error*-last*error*kiri) > 100. Jika ya akan melanjutkan ke urutan r, jika tidak akan melanjutkan ke urutan t.
r	Isi variabel pwmkiri dengan nilai 1152. Setelah itu,

	melanjutkan ke urutan t.
s	Isi variabel pwmkiri dengan nilai 1152. Setelah itu, melanjutkan ke urutan q.
t	Isi variabel last*error*kiri dengan nilai pada veriabel *error*.
u	Hasil dari ((*angle*+30)-*setpoint*+360) simpan ke variabel *error*.
v	Mengecek apakah variabel *error* > 360. Jika ya akan melanjutkan ke urutan w, jika tidak akan melanjutkan ke urutan x.
w	Hasil dari (*error*-360) simpan kembali pada variabel *error*.
x	Mengecek apakah variabel *error* > 360. Jika ya akan melanjutkan ke urutan y, jika tidak akan melanjutkan ke urutan z.
y	Hasil dari (*error*-360) akan disimpan kembali pada variabel *error*.
z	Hasil dari (KP**error*)+((*error*-last*error*kanan)*KD) simpan ke Variabel PD.
aa	Hasil dari ((KI**error*)+*error*totalkanan) simpan ke Variabel I.
ab	Mengecek apakah nilai (PD+I) > 32767) dan (PD+I) < (65535-150). Jika ya akan melanjutkan ke urutan ac, jika tidak akan melanjutkan ke urutan ad.
ac	Kurangi nilai (-150) dengan nilai pada variabel PD, lalu simpan hasilnya ke variabel *error*totalkanan. Setelah itu, melanjutkan ke urutan af.
ad	Mengecek apakah nilai (PD+I) > 150 dan (PD+I) <= 32767. Jika ya akan melanjutkan ke urutan ae, jika tidak akan melanjutkan ke urutan ag.
ae	Kurangi nilai (150) dengan niali pada variabel PD, lalu simpan hasilnya ke variabel *error*totalkanan. Setelah itu, melanjutkan ke urutan af.
af	Hasil dari (KI*(PD+*error*totalkanan+120))/30) simpan ke variabel I. setelah itu lanjutkan ke urutan ah.
ag	Isi variabel *error*totalkiri dengan nilai pada variabel I. Setelah itu, melanjutkan ke urutan af.
ah	Mengecek apakah nilai variabel *error* <= 210. Jika ya akan melanjutkan ke urutan ai, jika tidak akan melanjutkan ke urutan al.
ai	Tambal nilai pada variabel I dengan nilai 1150 lalu hasilnya simpan kedalam variabel pwmkanan.
aj	Mengecek apakah nilai (*Error*-last*error*kanan) > 100. Jika ya akan melanjutkan ke urutan ak, jika tidak akan melanjutkan ke urutan am.

ak	Isi variabel pwmkanan dengan nilai 1150. Setelah itu, melanjutkan ke urutan am.
al	Isi variabel pwmkanan dengan nilai 1150. Setelah itu, melanjutkan ke urutan aj.
am	Isi variabel last*error*kanan dengan nilai pada veriabel *error*.
an	Mengirimkan nilai pulsa ke *brushless* motor kiri.
ao	Mengirimkan nilai pulsa ke *brushless* motor kanan.
ap	Keluar dari prosedur dan kembali mengeksekusi intruksi selanjutnya pada program utama pada mikrokontroler dua.

3.3.3 Software Pendukung

Sofware merupakan pendukung dalam pembuatan program atau pembuatan *interface* untuk prototipe satelit. Dalam pembuatan program untuk prototipe satelit *software* pendukung yang digunakan adalah *basic stamp editor* v2.5 , sedangkan sofware pendukung untuk *interface* prototipe satelit adalah *visual basic* 6.0.

3.3.4 Deklarasi Variabel dan Tipe Data

Setiap program terdapat deklarasi variabel yang berguna untuk mendefinisikan pengenal yang akan digunakan di dalam program. Dalam pemrograman, suatu variabel harus memiliki kejelasan tentang data apa yang dapat disimpannya. Suatu variabel tidak bisa menyimpan semua jenis nilai atau data, melainkan hanya satu jenis tipe saja. Berikut deklarasi variabel dan tipe data pada perancangan prototipe satelit

Tabel 3.11. Deklarasi variabel dan tipe data pada mikrokontroler satu

Deklarasi variabel	Tipe Data
setpoint	word
x	word
y	word
angle	word
status	nib
Xaxel	byte
Yaxel	byte
f	byte

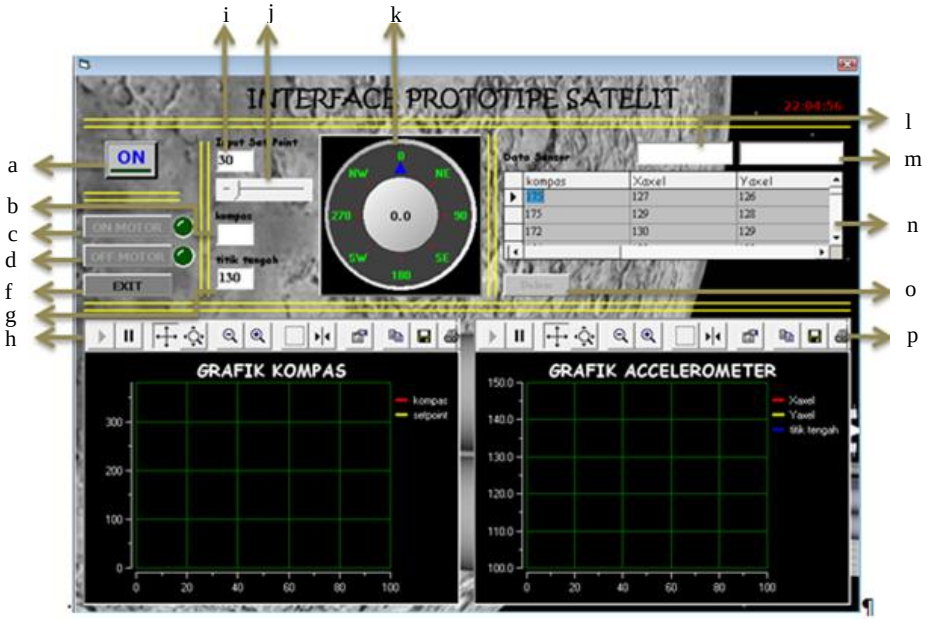
j	byte
stat	nib

Tabel 3.12. Deklarasi variabel dan tipe data pada mikrokontroler dua

Deklarasi variabel	Tipe Data
angle	word
motorkiri	word
motorkanan	word
error	word
*error*totalkanan	word
*error*totalkiri	word
last*error*kanan	word
last*error*kiri	word
PD	word
I	word
setpoint	word
f	byte
j	byte

3.3.5 Perancangan Interface di Visual Basic 6.0

Dalam perancangan *interface* untuk prototipe satelit menggunakan *visual basic* 6.0. *Interface* yang ditampilkan adalah sebagai berikut:



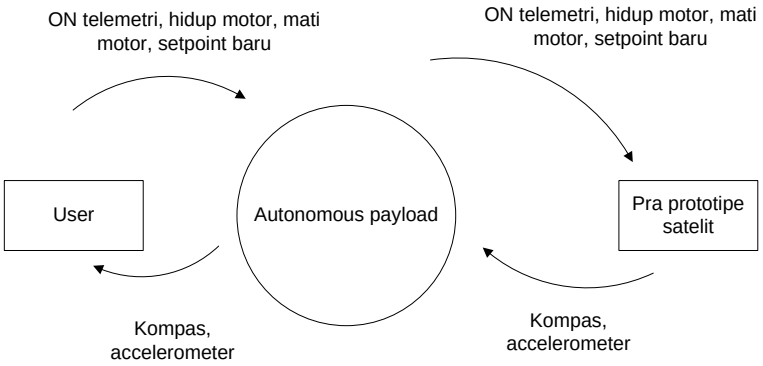
Gambar 3.18. *Interface* prototipe satelit

Tabel 3.13. Penjelasan *interface* prototipe satelit

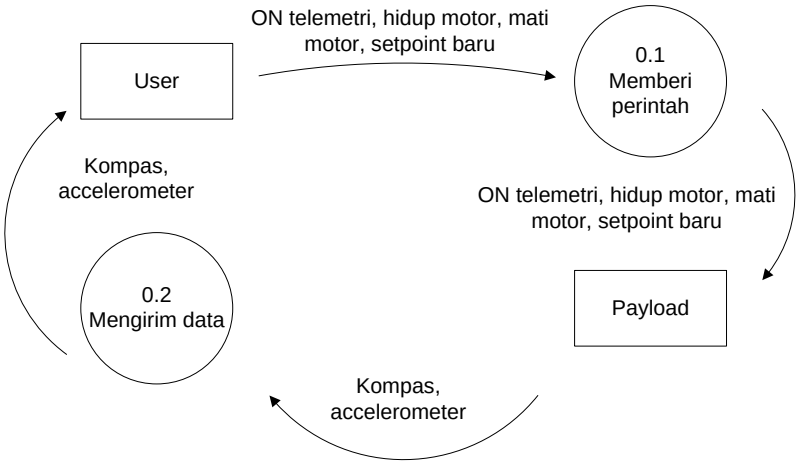
Urutan	Keterangan
a	Mengaktifkan port yang akan dipakai dan telemetri hidup
b	Data sensor kompas yang diterima dari prototipe satelit
c	Mengirimkan data untuk menghidupkan motor pada prototipe satelit dengan memberikan nilai pada variabel j di prototipe satelit.
d	Mengirimkan data untuk mematikan motor pada prototipe sateli dengan memberikan nilai pada variabel j di prototipe satelit
e	Keluar dari program interface prototipe satelit
f	Tampilan grafik untuk data sensor kompas yang di terima dari prototipe satelit
g	Garis yang merupakan referensi untuk mengejar kesetimbangan pada grafik *accelerometer.*
h	Tampilan *input setpoint*
i	*Slider* untuk mengubah *setpoint* baru pada prototipe satelit.
j	Data sensor kompas yang ditampilkan dalam bentuk kompas analog.
k	Semua data yang dikirimkan ke prototipe satelit dengan command “1e”
l	Data asli sensor kompas dan *accelerometer* yang diterima dari prototipe satelit Tampilan *database* untuk sensor kompas dan *accelerometer*.
m	Data sensor kompas dan accelerometer yang disimpan pada *datebase* dan ditampilkan pada tabel
n	Menghapus *database* yang sudah tersimpan.
o	Tampilan grafik untuk data sensor *accelerometer* yang diterima dari prototipe satelit.

3.3.6 Perancangan DFD (Data Flow Diagram)

DFD adalah salah satu alat pembuatan model yang sering digunakan, khususnya bila fungsi-fungsi sistem merupakan bagian yang lebih penting dan kompleks dari pada data yang dimanipulasi oleh sistem. Berikut adalah DFD utuk *interface* prototipe satelit :



Gambar 3.19. Diagram konteks



Gambar 3.20. Diagram level 0