BAB III
PERANCANGAN SISTEM

Bab ini akan membahas tentang perancangan sistem pengiriman data pengukuran parameter meteorologi serta pencarian posisi terhadap arah gerak *payload (seeking)*. Pada perancangan sistem ini dibagi menjadi tiga bagian penting, yaitu:

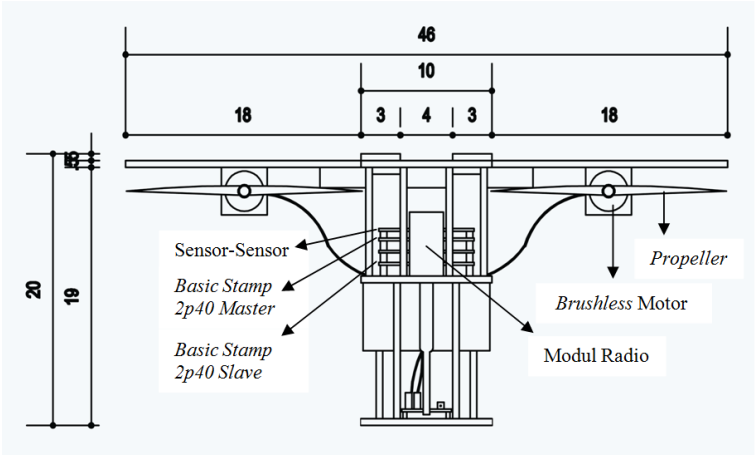
- Perancangan mekanik,
- Perancangan perangkat keras, dan
- Perancangan perangkat lunak.

3.1 Perancangan Mekanik

Perancangan mekanik *payload*, dirancang berdasarkan ukuran dan berat agar pencarian posisi terhadap arah gerak *payload* dapat tercapai. Ukuran dan berat *payload* yang dirancang adalah sebagai berikut:

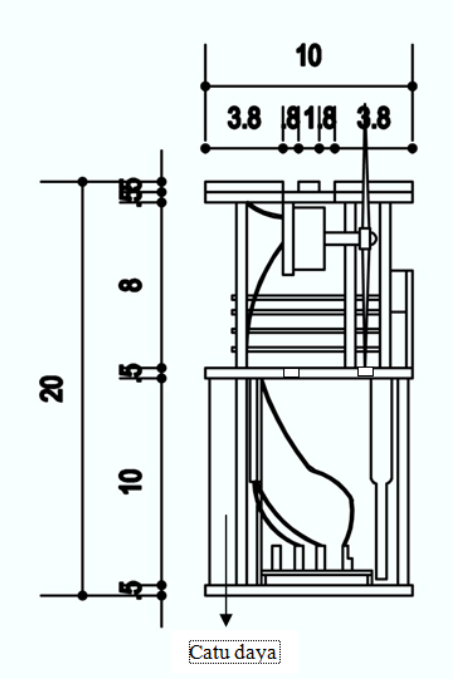
- Diameter *Payload* : (100 ± 1) mm
- Tinggi *Payload* : (200 ± 1) mm
- Berat *Payload* : (1000 ± 10) g

Gambar 3.1 menunjukkan *payload* tampak depan dengan kondisi sayap terbuka.



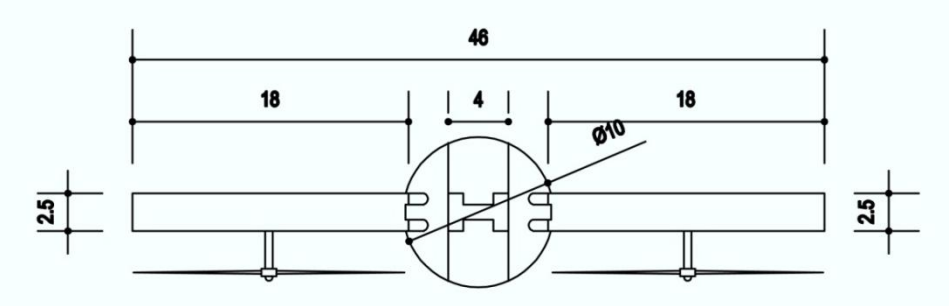
Gambar 3.1 Payload tampak depan dengan kondisi sayap terbuka

Gambar 3.2 menunjukkan *payload* tampak samping dengan kondisi sayap terbuka.



Gambar 3.2 Payload tampak samping dengan kondisi sayap terbuka

Gambar 3.3 menunjukkan *payload* tampak atas dengan kondisi sayap terbuka.



Gambar 3.3 Payload tampak atas dengan kondisi sayap terbuka

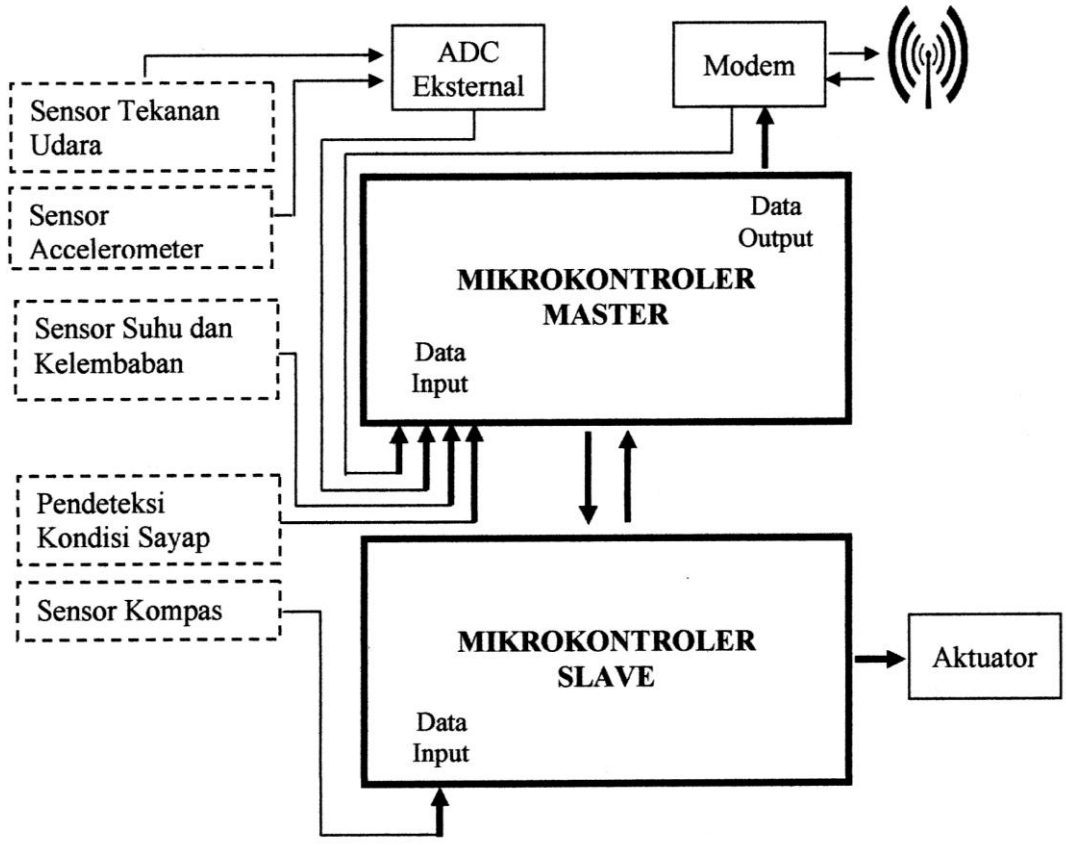
Penempatan komponen:

- *Brushless* motor dan *propeller* dipilih sebagai aktuator agar *payload* dapat melakukan pencarian posisi terhadap arah gerak *payload*. Aktuator ini ditempatkan pada bagian kanan dan kiri sayap *payload* agar memudahkan pada saat melakukan pencarian posisi (*seeking*).

- Sensor-sensor ditempatkan pada bagian tengah *payload*. Khusus untuk sensor suhu ditempatkan terpisah dari komponen-komponen elektronika yang dapat menimbulkan panas untuk menjamin keakuratan data suhu udara yang terukur di sekitar *payload*.
- Mikrokontroler *master* diletakkan pada bagian bawah sensor-sensor.
- Mikrokontroler *slave* diletakkan pada bagian bawah mikrokontroler *master*.
- Catu daya ditempatkan di bagian belakang *payload*.
- Modul radio ditempatkan pada bagian depan *payload* beserta antenanya yang dipasang secara *vertikal*.

3.2 Perancangan Perangkat Keras

Diagram blok sistem yang akan dibuat ditunjukkan pada gambar 3.4.



Gambar 3.4 Diagram blok sistem payload

Berikut penjelasan blok-blok yang terdapat pada gambar 3.4.

a. Mikrokontroler Master

Bagian blok ini memuat Mikrokontroler *Basic Stamp 2p40*. Bagian blok ini berfungsi sebagai pengolah data yang diterima dari sensor tekanan udara, sensor *accelerometer*, sensor suhu dan kelembaban. Blok ini juga berfungsi sebagai pengatur komunikasi data untuk pengiriman data telemetri.

b. Mikrokontroler Slave

Bagian blok ini memuat Mikrokontroler *Basic Stamp 2p40*. Bagian blok ini berfungsi sebagai pengolah data yang diterima dari sensor kompas dan juga berfungsi sebagai pengendali arah gerak *payload*.

c. Sensor Tekanan Udara

Bagian blok ini memuat sensor MPXA6115A6U. Bagian blok ini berfungsi sebagai pengukur tekanan di udara yang kemudian datanya akan diolah pada mikrokontroler *master*.

d. Sensor Accelerometer

Bagian blok ini memuat sensor MMA3201EG. Bagian blok ini berfungsi sebagai pengukur percepatan geravitasi yang dialami *payload* yang kemudian datanya akan diolah pada mikrokontroler *master*.

e. Sensor Suhu dan Kelembaban

Bagian blok ini memuat sensor SHT75. Bagian blok ini berfungsi sebagai pengukur suhu dan kelembaban di sekitar *payload* yang kemudian datanya akan diolah pada mikrokontroler *master*.

f. Sensor Kompas

Bagian blok ini memuat sensor hitachi HM55B. Bagian blok ini berfungsi untuk mengetahui arah kompas atau arah gerak *payload* yang kemudian datanya akan diolah pada mikrokontroler *slave.*

g. Pendeteksi Kondisi Sayap

Bagian blok ini memuat rangkaian *push button* yang dibuat sebagai pendeteksi kondisi sayap.

h. Modem

Blok ini memuat modul radio YS-1020UB yang berfungsi mengubah sinyal informasi ke dalam sinyal pembawa (*carrier*) dan siap untuk dikirimkan, serta memisahkan sinyal informasi (yang berisi data atau pesan) dari sinyal pembawa (*carrier*) yang diterima sehingga informasi tersebut dapat diterima dengan baik.

i. Aktuator

Bagian blok ini memuat *brushless* motor dan *propeller*. Bagian blok ini berfungsi sebagai penggerak *payload.* Blok ini dikendalikan oleh mikrokontroler *slave.*

3.2.1 Mikrokontroler (Basic Stamp 2p40)

Mikrokontroler yang digunakan pada perancangan *payload* ini adalah mikrokontroler jenis *basic stamp 2P40* dengan 32 *port* yang dapat digunakan sebagi *input* atau *output*. Pemilihan mikrokontroler jenis ini didasari kemampuannya yang cukup handal dan pemrograman yang tidak terlalu sulit. Pada perancangan *payload* yang dibuat, penulis menggunakan dua buah mikrokontroler *basic stamp 2p40,* yaitu sebagai mikrokontroler *master* dan mikrokontroler *slave.*

Berikut ini merupakan konfigurasi *port* yang digunakan mikrokontroler *master*, seperti yang terlihat pada tabel 3.1.

Tabel 3.1 Konfigurasi port yang digunakan pada mikrokontroler master.

Nama Pin	No. Pin	Fungsi	Keterangan
MAIN I/O	0	*Output*	Penerima (RX) ke RF
MAIN I/O	2	*Input*	Pengirim (TX) dari RF
MAIN I/O	12	*Output*	SHT75 (CLK)
MAIN I/O	14	*Input*	SHT75 (Data)
AUX I/O	5	*Output*	Pengirim (TX) ke mikrokontroler *slave*
AUX I/O	6	*Output*	ADC 0832 (Cs)
AUX I/O	7	*Input*	Penerima (RX) dari mikrokontroler *slave*
AUX I/O	8	*Output*	ADC 0832 (CLK)
AUX I/O	10	*Output*	ADC 0832 (Dout)
AUX I/O	12	*Input*	ADC 0832 (Din)
AUX I/O	14	*Input*	Data pendeteksi sayap

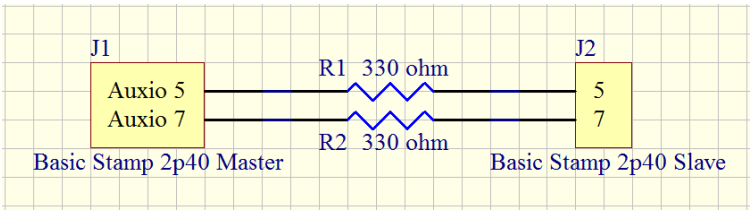
Berikut ini merupakan konfigurasi *port* yang digunakan mikrokontroler *slave*, seperti yang terlihat pada tabel 3.2.

Tabel 3.2 Konfigurasi port yang digunakan pada mikrokontroler slave.

Nama Pin	No. Pin	Fungsi	Keterangan
MAIN I/O	5	*Input*	Penerima (RX) dari mikrokontroler *master*
MAIN I/O	7	*Output*	Pengirim (TX) ke mikrokontroler master
MAIN I/O	10	*Output*	HM55B (CLK)
MAIN I/O	12	*Output*	HM55B (En)
MAIN I/O	14	*Input-Output*	HM55B (Din dan Dout)
AUX I/O	1	*Output*	Motor *brushless* kanan
AUX I/O	3	*Output*	Motor *brushless* kiri

Mikrokontroler *master* berfungsi untuk mengolah data dan sebagai pengatur pengiriman data *(telemetri)*, sedangkan mikrokontroler *slave* berfungsi untuk membaca data pengukuran sensor kompas dan pengendali arah gerak *payload.* Dua buah mikrokontroler ini akan saling berkomunikasi satu sama lain. Mikrokontroler *master* meminta hasil data pengukuran sensor kompas kepada mikrokontroler *slave* dan secara bersamaan mengirimkan data *set point* serta memberi tahu kapan *payload* siap untuk

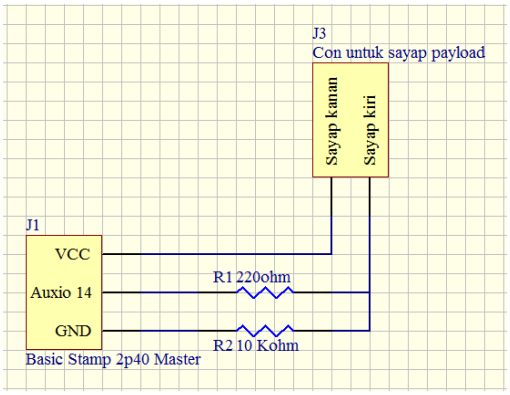
dikendalikan. Rangkaian komunikasi mikrokontroler *master* dengan mikrokontroler *slave*, seperti yang terlihat pada gambar 3.5.



Gambar 3.5 Rangkaian komunikasi mikrokontroler master dengan mikrokontroler slave

3.2.2 Pendeteksi Sayap (Rangkaian Push button)

Mikrokontroler *master* melakukan pengecekan kondisi pada sayap dengan membaca nilai *output* dari rangkaian yang dibuat untuk mengetahui kondisi sayap. Pada saat kondisi salah satu atau kedua sayap tertutup maka rangkaian ini akan memberikan nilai 0 *(low)*. Jika kondisi kedua sayap terbuka maka rangkaian ini akan memberikan nilai 1 *(high)* pada mikrokontroler *master*, hasil nilai pemeriksaan yang dilakukan mikrokontroler *master* akan dikirim ke mikrokontroler *slave* sebagai perintah kapan *payload* siap untuk dikendalikan. Mikrokontroler *slave* akan melakuan pengendalian arah pada *payload* sesuai dengan *setpoint* yang diberikan oleh mikrokontroler *master* dengan memanfaatkan data pengukuran kompas sebagai acuan. Rangkaian yang dibuat sebagai pendeteksi sayap, ditunjukkan pada gambar 3.6.



Gambar 3.6 Rangkaian pendeteksi sayap

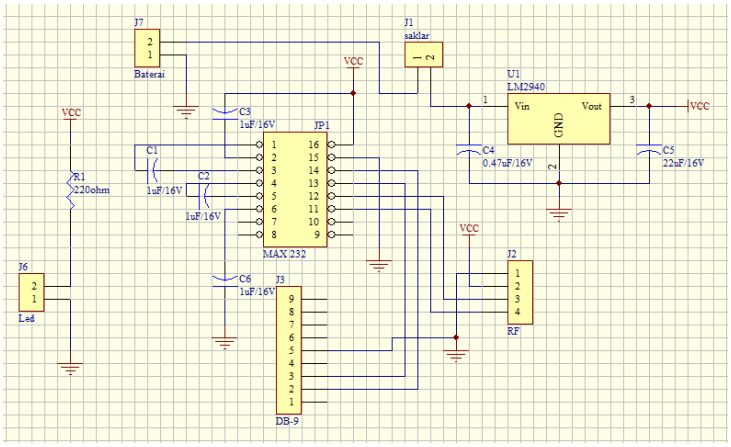
Tabel 3.3 menunjukkan *output* yang diinginkan sesuai kodisi sayap.

Tabel 3.3 Tabel kebenaran untuk kondisi sayap.

Sayap Kiri	Sayap Kanan	Output
Tertutup	Tertutup	*Low* (0)
Tertutup	Terbuka	*Low* (0)
Terbuka	Tertutup	*Low* (0)
Terbuka	Terbuka	*High* (1)

3.2.3 Pengubah Level Digital ke RS232 (MAX232)

Rangkaian RS232 berfungsi untuk komunikasi antara modem dengan komputer. Komponen utama menggunakan IC MAX232, yaitu sebuah IC yang dapat mengubah format *digital* ke dalam sebuah format atau *level* RS232, dimana pada *level* RS232, tegangan *high* diwakili dengan tegangan +3 sampai +25 V. Diantara -3 dan +3 merupakan tegangan *invalid*. Rangkaian RS232 ditunjukkan pada gambar 3.7.



Gambar 3.7 Rangkaian RS232

3.2.4 Pengubah Sinyal Analog ke Digital (ADC0832)

ADC yang digunakan pada perancangan *payload* ini, yaitu menggunakan ADC0832. Pemilihan ADC jenis ini disesuaikan dengan kebutuhan perancangan *payload*. Pada perancangan *payload* ini, memerlukan dua *channel* sinyal *input* ADC yang digunakan untuk membaca

sensor *accelerometer* dan sensor tekanan udara.

ADC0832 memiliki tegangan *referensi* (V_{REF}) yang terhubung langsung dengan tegangan sumber (Vcc), sehingga penggunaannya lebih sederhana karena dapat langsung digunakan tanpa adanya rangkaian tambahan.

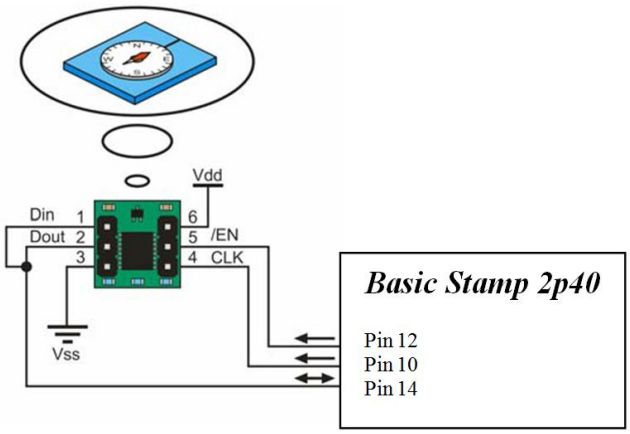
Channel 0 (pin 2) digunakan sebagai sinyal *input* untuk membaca data keluaran dari sensor *accelerometer* dan *channel* 1 (pin 3) digunakan sebagai sinyal *input* untuk membaca data keluaran dari sensor tekanan udara. Pada dasarnya *Analog To Digital Converter* (ADC) memiliki 2 bagian, yaitu bagian *multiplexer* dan bagian *converter*. Bagian multiplexer ini mempunyai 2 buah masukan, setiap masukan memilki alamat sendiri sehingga dapat dipilih secara terpisah melalui *address* A0 dan A1. Tabel 3.4 menunjukkan alamat *multiplexer* dari masing-masing masukan *channel*.

Tabel 3.4 MUX Addressing: ADC0832 Single-Ended MUX Mode

MUX address		**Channel**	
SGL/$\overline{\text{DIF}}$	ODD/SIGN	0	1
1	0	+	
1	1		+

3.2.5 Sensor Kompas (Hitachi HM55B)

Pada perancangan *payload* ini menggunakan hitachi HM55B sebagai sensor kompas. Pemilihan sensor ini didasari kemampuannya yang cukup handal dan kemampuan pembacaan data yang cepat, hanya 30 sampai 40 ms antara pengukuran awal dan data siap dibaca kembali. Tentunya pembacaan data yang cepat ini sangat dibutuhkan agar pencarian arah gerak *payload* dapat tercapai dengan baik. Penggunaan modul hitachi HM55B ini juga sangat sederhana karena tidak memerlukan rangkaian tambahan yang rumit. Rangkaian komunikasi modul hitachi HM55B dengan *Basic Stamp*, diperlihatkan pada gambar 3.8.



Gambar 3.8 Rangkaian komunikasi modul hitachi HM55B dengan Basic Stamp 2p40

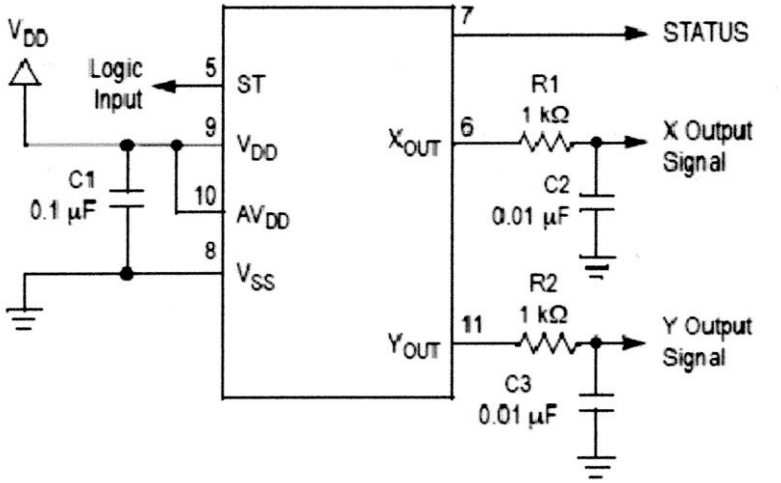
Modul hitachi HM55B akan mengirimkan data axis x dan axis y, kedua data kedua axis ini akan diolah oleh mikrokontroler *slave* yang kemudian diubah dalam satuan derajat dengan menggunakan persamaan di bawah ini.

$$Sudut\ \theta = \arctan \frac{-(\text{axis y})}{(\text{axis x})} \qquad (3.1)$$

3.2.6 Sensor Percepatan (MMA3201EG)

Pada perancangan *payload* ini menggunakan sensor MMA3201EG yang digunakan sebagai sensor *accelerometer,* yaitu pengukur data percepatan gravitasi yang terjadi pada *payload.*

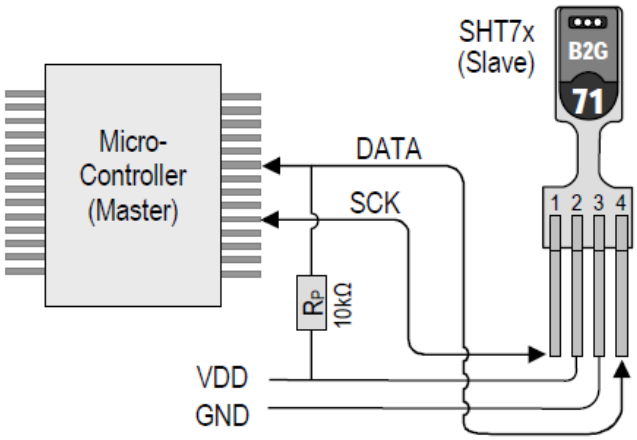
Pemilihan sensor ini didasari kemampuannya yang mempunyai sensitivitas hingga 40g pada sumbu X dan sumbu Y, dimana 1(satu) g = 9,8 m/s^2. Rangkaian sensor MMA3201EG yang digunakan pada perancangan *payload* seperti yang terlihat pada gambar 3.9.



Gambar 3.9 Rangkaian sensor MMA3201EG

3.2.7 Sensor Suhu dan Kelembaban (SHT75)

Pada perancangan *payload* ini menggunakan sensor SHT75 yang digunakan untuk pengukur suhu dan kelembaban di sekitar *payload.* Pemilihan sensor ini didasari kemampuannya yang mempunyai tingkat akurasi data yang baik, yaitu untuk akurasi kelembaban adalah ±1,8 %RH dan untuk akurasi suhunya adalah 0.3 ^{o}C. Data *output* dari sensor SHT75 ini berupa data *digital*, sehingga penggunaannya tidak terlalu sulit. Selanjutnya data *output* ini akan diolah oleh mikrokontroler *master.* Rangkaian komunikasi sensor SHT75 dengan *basic stamp* ditunjukkan pada gambar 3.10.



Gambar 3.10 Rangkaian komunikasi sensor SHT75 dengan Basic Stamp

Sensor SHT75 akan mengirimkan data *output* berupa data suhu dan kelembaban yang akan diolah oleh mikrokontroler *master*. Data suhu tersebut akan diubah dalam satuan derajat *celcius* (^{0}C) dengan menggunakan persamaan 3.2, sedangkan data kelembaban diubah ke dalam satuan %RH dengan menggunakan persamaan 3.3.

$$T = -40.1 + 0.01 * SO_T \quad (3.2)$$

$$RH_{Linier} = -2.0468 + 0.0367 * SO_{RH} + -1.5955x10^{-6} * SO_{RH}{}^2 \quad (3.3)$$

Keterangan :

RH_{Linier} = Kelembaban dalam satuan %RH

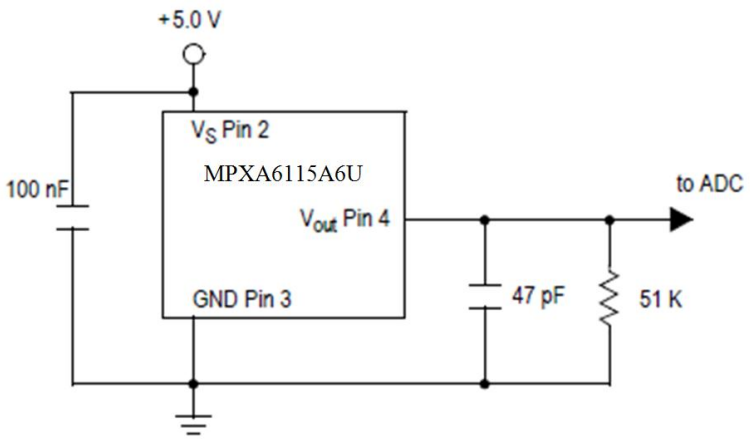
SO_{RH} = data kelembaban yang terukur dari sensor SHT75

T = Suhu dalam satuan derajat *celcius* (^{0}C)

SO_T = data suhu yang terukur dari sensor SHT75

3.2.8 Sensor Tekanan Udara (MPXA6115A6U)

Pada perancangan *payload* ini menggunakan sensor MPXA6115A6U yang digunakan untuk pengukuran tekanan di udara. Pemilihan sensor ini didasari kemampuannya yang mempunyai tingkat *error* yang kecil, yaitu 1,5%. Rangkaian sensor MPXA6115A6U yang digunakan pada perancangan alat yang dibuat, diperlihatkan pada gambar 3.11.



Gambar 3.11 Rangkaian sensor MPXA6115A6U

Data *output* dari sensor MPXA6115A6U berupa data analog. Data ini akan diubah ke dalam desimal dengan ADC, kemudian diberikan ke mikrokontroler *master* yang hasilnya diubah ke dalam satuan kilo *pascal* (kPa) dengan menggunakan persamaan 3.4.

$$V_{out} = V_s * \ 0.009 \ * P - 0.095$$

$$P = \frac{\frac{V_{out}}{V_s} + 0.095}{0.009}$$

$$P = \frac{\frac{\frac{output\ ADC}{255} x\ 5v}{5v} + 0.095}{0.009}$$

$$P = \frac{output\ ADC}{255 * 0.009} + \frac{0.095}{0.009} \qquad (3.4)$$

3.2.9 Radio Frekuensi (Modul Radio YS-1020UB)

Media komunikasi yang digunakan pada perancangan alat yang dibuat, yaitu melalui frekuensi radio (RF) dengan menggunakan modul radio YS-1020UB. Modul radio YS-1020UB mempunyai 8 kanal dengan frekuensi yang berbeda. Frekuensi yang digunakan pada *payload* ini adalah 433 Mhz pada kanal 6. Jarak jangkauan komunikasi maksimal sekitar 800 meter pada *baudrate* 9600 bps.

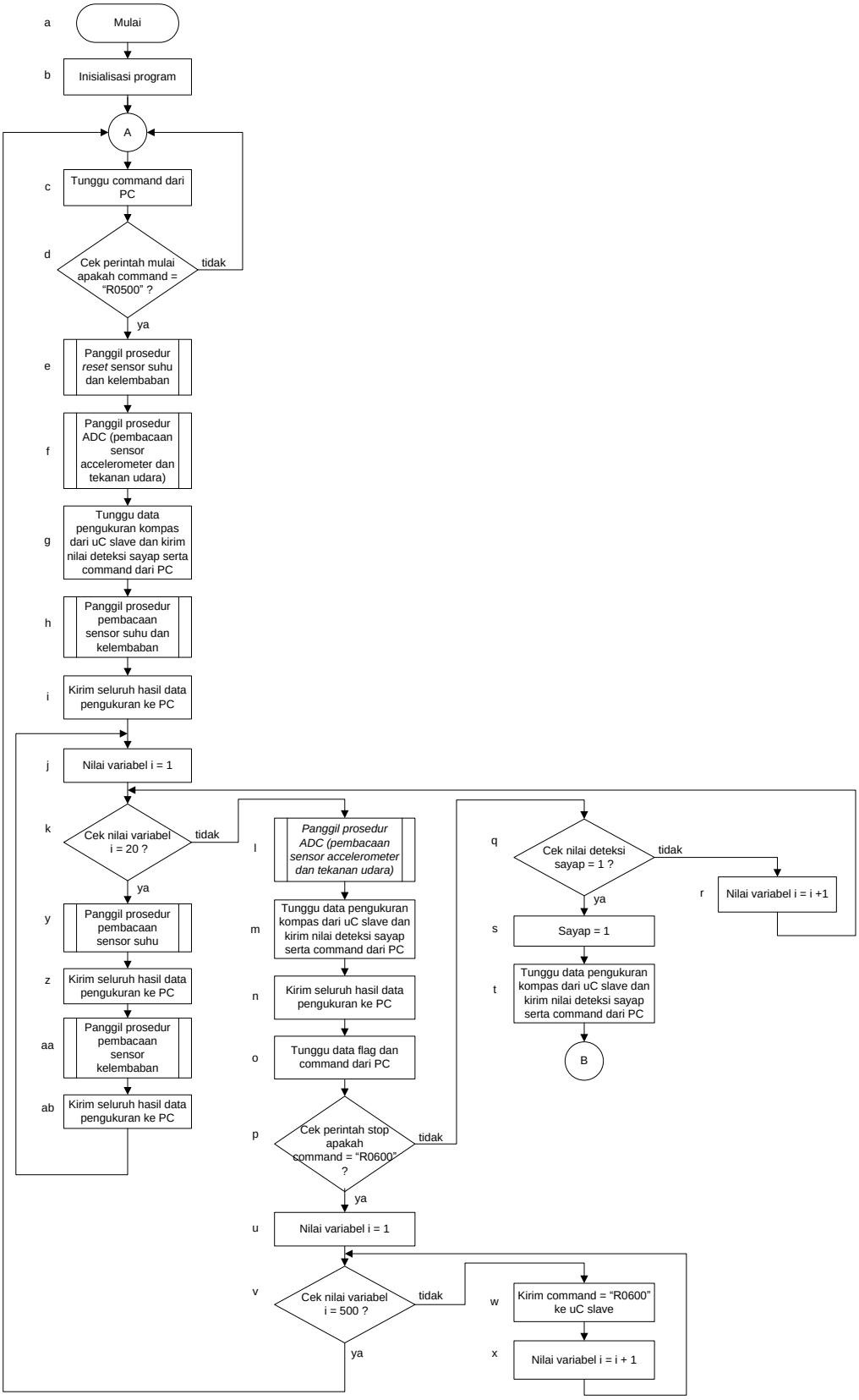
Dari konfigurasi pin-pin radio YS-1020UB, pin yang digunakan pada perancangan *payload* hanya terdiri dari GND, Vcc, RXD/TTL dan TXD/TTL yang dihubungkan langsung pada pin di mikrokontroler.

3.3 Perancangan Perangkat Lunak

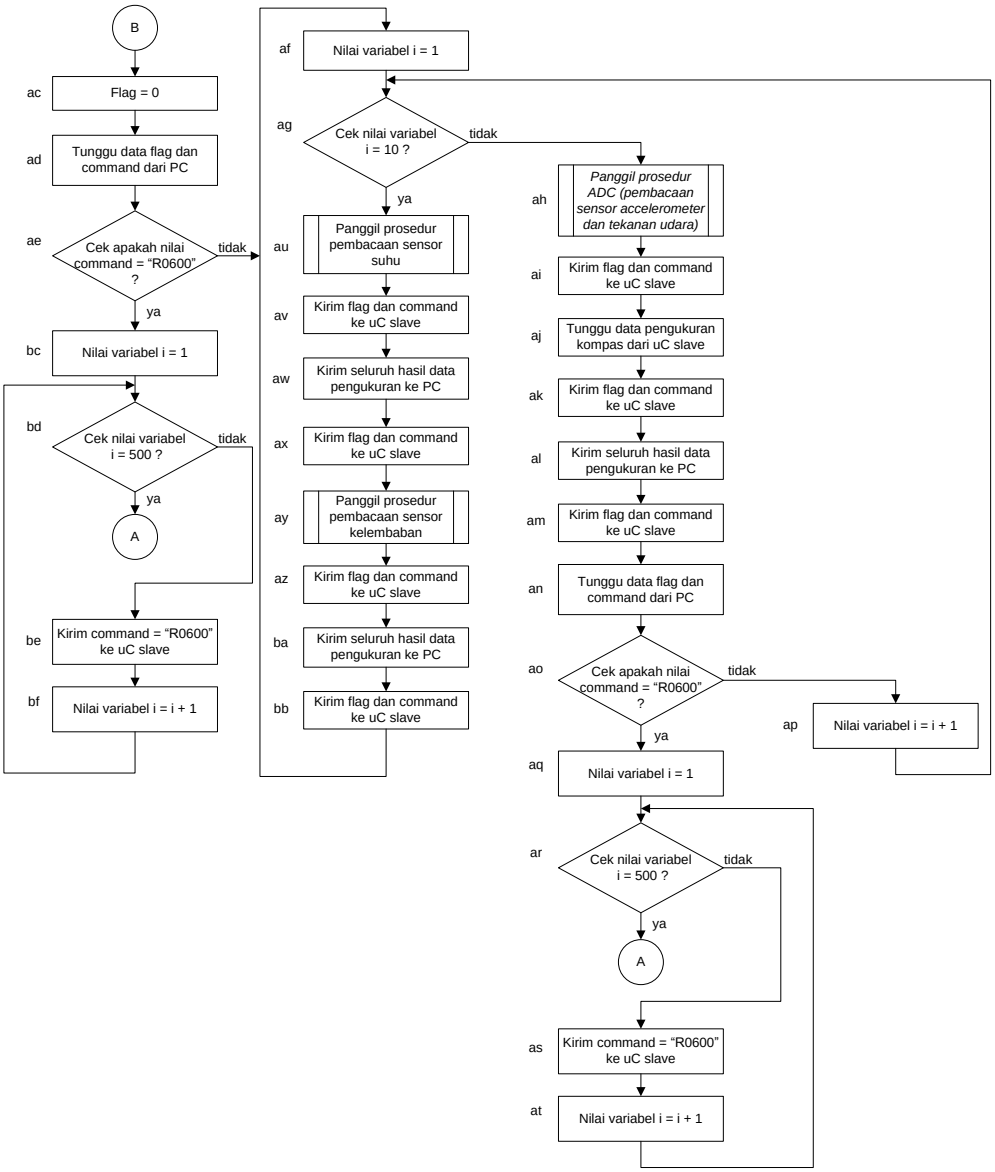
Setelah dalam pembuatan perangkat keras selesai, bagian yang paling penting dalam pembuatan tugas akhir ini yaitu dalam merancang suatu perangkat lunak. Dalam tugas akhir ini, perancangan perangkat lunak dibagi menjadi dua bagian yaitu:

- Perancangan perangkat lunak untuk mikrokontroler *master*.
- Perancangan perangkat lunak untuk mikrokontroler *slave*.

3.3.1 Diagram Alir Program Mikrokontroler Master



Gambar 3.12 Diagram alir program mikrokontroler master



Gambar 3.13 Diagram alir program mikrokontroler master (lanjutan)

Berikut ini penjelasan diagram alir program pada mikrokontroler *master*:

a. Memulai awal program.
b. Pendeklarasian program. Pengaturan nama prosesor, nama kompiler, konfigurasi pin-pin dan deklarasi variabel.
c. Tunggu data *command* dari PC.
d. Pemeriksaan apakah data yang dikirim PC merupakan perintah untuk memulai eksekusi seluruh program, jika data *command* = ’R0500’ maka lanjutkan ke poin e, jika tidak maka eksekusi akan kembali dilakukan pada poin c.
e. Panggil prosedur untuk mengatur ulang perangkat koneksi sensor suhu dan kelembaban.
f. Panggil prosedur pembacaan ADC, yaitu data pengukuran sensor *accelerometer* dan sensor tekanan udara.
g. Tunggu data hasil pengukuran sensor kompas dari mikrokontroler *slave* dan mengirimkan data deteksi sayap serta data *command* dari PC.
h. Panggil prosedur pembacaan sensor suhu dan kelembaban.
i. Memberikan perintah untuk mengirim seluruh hasil data pengukuran sensor-sensor ke PC.
j. Isi nilai pada variabel i dengan nilai 1.
k. Pemeriksaan apakah nilai pada variabel i = 20, jika ya lanjut ke poin y, jika tidak lanjut ke poin l.
l. Panggil prosedur pembacaan ADC, yaitu data pengukuran sensor *accelerometer* dan sensor tekanan udara.
m. Tunggu data hasil pengukuran sensor kompas dari mikrokontroler *slave* dan mengirimkan data deteksi sayap serta data *command* dari PC.
n. Memberikan perintah untuk mengirim seluruh hasil data pengukuran sensor-sensor ke PC.
o. Tunggu data *flag* dan *command* dari PC.
p. Pemeriksaan apakah data *command* yang dikirim PC merupakan perintah untuk menghentikan eksekusi, jika data *command* =

‘R0600’ maka lanjutkan ke poin u, jika tidak maka lanjutkan ke poin q.

q. Pemeriksaan nilai deteksi sayap apakah bernilai 1, jika bernilai 1 maka lanjutkan ke poin s, jika tidak lanjutkan ke poin r.
r. Jumlahkan nilai pada variabel i dengan 1 dan hasil penjumlahan simpan kembali pada variabel i, kemudian lanjutkan ke poin j.
s. Isi nilai pada variabel sayap dengan nilai 1.
t. Tunggu data hasil pengukuran sensor kompas dari mikrokontroler *slave* dan mengirimkan data deteksi sayap serta data *command* dari PC.
u. Isi nilai pada variabel i dengan nilai 1.
v. Pemeriksaan apakah nilai pada variabel i = 500, jika ya kembali ke poin c, jika tidak lanjutkan ke poin w.
w. Memberikan perintah untuk mengirim data *command,* yaitu ’R0600’ ke mikrokontroler *slave*.
x. Jumlahkan nilai pada variabel i dengan nilai 1 dan hasil penjumlahan simpan kembali pada variabel i, kemudian lanjutkan ke poin u.
y. Panggil prosedur pembacaan data suhu.
z. Mengirim seluruh hasil data pengukuran sensor-sensor ke PC.
aa. Panggil prosedur pembacaan data kelembaban.
ab. Memberikan perintah untuk mengirim seluruh hasil data pengukuran sensor-sensor ke PC, setelah data diterima kembali ke poin i.
ac. Isi nilai pada variabel flag dengan nilai 0.
ad. Tunggu data *flag* dan *command* dari PC.
ae. Pemeriksaan apakah data *command* yang dikirim PC merupakan perintah untuk menghentikan eksekusi, jika data *command* = ‘R0600’ maka lanjutkan ke poin bc, jika tidak maka lanjutkan ke poin af.
af. Isi nilai pada variabel i dengan niali 1.

ag. Pemeriksaan apakah nilai pada variabel i = 10, jika ya lanjutkan ke poin au, jika tidak lanjutkan ke poin ah.

ah. Panggil prosedur pembacaan ADC, yaitu data pengukuran sensor *accelerometer* dan sensor tekanan udara.

ai. Memberikan perintah untuk mengirim data variabel *flag* dan *command* pada mikrokontroler *slave*.

aj. Tunggu data hasil pengukuran sensor kompas dari mikrokontroler *slave.*

ak. Memberikan perintah untuk mengirim data variabel *flag* dan *command* pada mikrokontroler *slave*.

al. Memberikan perintah untuk mengirim seluruh hasil data pengukuran sensor-sensor ke PC.

am. Memberikan perintah untuk mengirim data variabel *flag* dan *command* pada mikrokontroler *slave*.

an. Tunggu data *flag* dan *command* dari PC.

ao. Pemeriksaan apakah data *command* yang dikirim PC merupakan perintah untuk menghentikan eksekusi, jika data *command* = ‘R0600’ maka lanjutkan ke poin aq, jika tidak maka lanjutkan ke poin ap.

ap. Jumlahkan nilai pada variabel i dengan nilai 1 dan hasil penjumlahan simpan kembali pada variabel i, kemudian lanjutkan ke poin af.

aq. Isi nilai pada variabel i dengan nilai 1.

ar. Pemeriksaan apakah nilai pada variabel i = 500, jika ya kembali ke poin c, jika tidak lanjutkan ke poin as.

as. Memberikan perintah untuk mengirim data *command,* yaitu ’R0600’ ke mikrokontroler *slave*.

at. Jumlahkan nilai pada variabel i dengan nilai 1 dan hasil penjumlahan simpan kembali pada variabel i, kemudian lanjutkan ke poin aq.

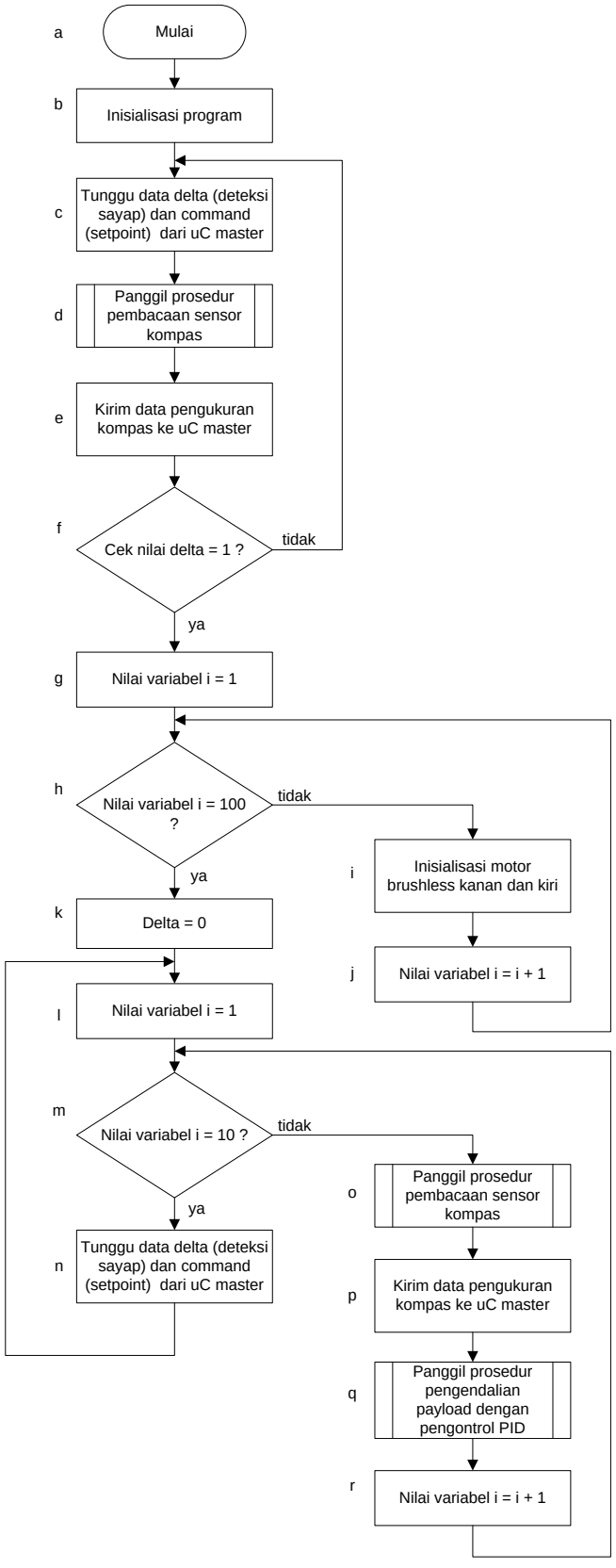
au. Panggil prosedur pembacaan data suhu.

av. Memberikan perintah untuk mengirim data variabel *flag* dan *command* pada mikrokontroler *slave*.
aw. Memberikan perintah untuk mengirim seluruh hasil data pengukuran sensor-sensor ke PC.
ax. Memberikan perintah untuk mengirim data variabel *flag* dan *command* pada mikrokontroler *slave*.
ay. Panggil prosedur pembacaan data kelembaban.
az. Memberikan perintah untuk mengirim data variabel *flag* dan *command* pada mikrokontroler *slave*.
ba. Memberikan perintah untuk mengirim seluruh hasil data pengukuran sensor-sensor ke PC.
bb. Memberikan perintah untuk mengirim data variabel *flag* dan *command* pada mikrokontroler *slave*.
bc. Isi nilai pada variabel i dengan nilai 1.
bd. Pemeriksaan apakah nilai pada variabel i = 500, jika ya kembali ke poin c, jika tidak lanjutkan ke poin be.
be. Memberikan perintah untuk mengirim data *command,* yaitu ’R0600’ ke mikrokontroler *slave*.
bf. Jumlahkan nilai pada variabel i dengan nilai 1 dan hasil penjumlahan simpan kembali pada variabel i, kemudian lanjutkan ke poin bc.

Diagram alir di atas adalah program untuk mikrokontroler *master* pada saat menerima data dari komputer (PC). Data *string* ’R0500’ yang diberikan pada mikrokontroler *master* digunakan untuk memulai jalannya seluruh instruksi program, dan data *string* ’R0600’ digunakan untuk menghentikan jalannya program kemudian siap untuk dijalankan kembali *(stanby)*. Data karakter yang diikuti dengan data desimal, seperti ’R’0030 digunakan untuk memberikan nilai *set point* 30^{o}, data ’R’0090 digunakan untuk memberikan nilai *set point* 90^{o}, data ’R’0180 digunakan untuk memberikan nilai *set point* 180^{o} dan data ’R’0270 digunakan untuk memberikan nilai *set point* 270^{o}.

Karakter bagian depan data, yaitu huruf ’R’ dibuat sebagai inisialisasi awal data, jika komputer (PC) mengirimkan data tidak didahului dengan huruf ’R’ maka pada mikrokontroler *master* data tidak diterima karena dianggap bukan data dari komputer (PC) pengendali *payload.* Satu digit data desimal setelah karkter ’R’, merupakan nilai untuk variabel *flag* dan tiga digit data desimal berikutnya merupakan nilai data untuk variabel *command,* nilai tiga digit data desimal ini akan dikirim oleh mikrokontroler *master* ke mikrokontroler *slave.* Data ini akan diproses pada mikrokontroler *slave* sebagai nilai *set point* yang akan dibandingkan dengan data hasil pengukuran sensor kompas.

3.3.2 Diagram Alir Program Mikrokontroler Slave



Gambar 3.14 Diagram alir program mikrokontroler slave

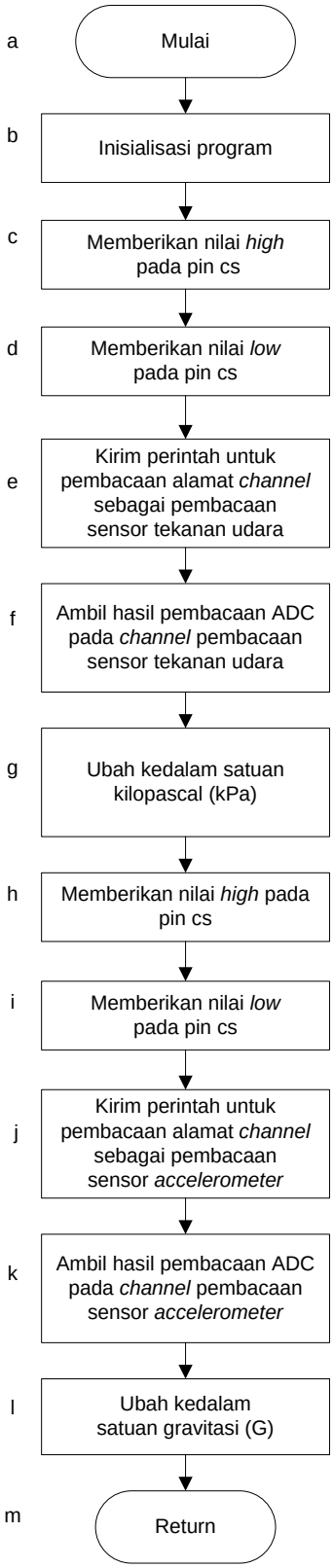
Berikut ini penjelasan diagram alir program pada mikrokontroler *slave*:

a. Memulai awal program.
b. Pendeklarasian program. Pengaturan nama prosesor, nama kompiler, konfigurasi pin-pin dan deklarasi variabel.
c. Tunggu data delta (deteksi sayap) dan *command (setpoint)* dari mikrokontroler *master*.
d. Panggil prosedur pembacaan sensor kompas.
e. Memberikan perintah untuk mengirimkan hasil data pengukuran sensor kompas ke mikrokontroler *master*.
f. Pemeriksaan apakah nilai pada variabel delta (deteksi sayap) = 1, jika ya lanjutkan ke poin g, jka tidak kembali ke poin c.
g. Isi nilai pada variabel i dengan nilai 1.
h. Pemeriksaan apakah nilai pada variabel i = 100, jika ya lanjutkan ke poin k, jika tidak lanjutkan ke poin i.
i. Inisialisasi motor *brushless* kanan dan kiri.
j. Jumlahkan nilai pada variabel i dengan nilai 1, kemudian hasil penjumlahan simpan kembali pada variabel i, kemudian kembali ke poin h.
k. Isi nilai pada variabel delta dengan nilai 0.
l. Isi nilai pada variabel i dengan nilai 1.
m. Pemeriksaan apakah nilai pada variabel i = 10, jika ya lanjutkan ke poin n, jika tidak lanjutkan ke poin o.
n. Tunggu data delta (deteksi sayap) dan *command (setpoint)* dari mikrokontroler *master,* setelah data diterima kembali ke poin l.
o. Panggil prosedur pembacaan sensor kompas.
p. Memberikan perintah untuk mengirimkan hasil data pengukuran sensor kompas ke mikrokontroler *master*.
q. Panggil prosedur untuk pengontrolan kendali *payload* dengan menggunakan pengontrol PID.
r. Jumlahkan isi nilai variabel i dengan nilai 1 dan hasil penjumlahan simpan kembali pada variabel i, kemudian kembali ke poin m.

Diagram alir pada gambar 3.14 adalah program untuk mikrokontroler *slave* pada saat menerima data dari mikrokontroler *master*. Mikrokontroler *master* akan mengirimkan data pada mikrokontroler *slave*, setelah data diterima oleh mikrokontroler *slave*, maka mikrokontroler *slave* akan menjalankan instruksi program, namun tidak semua instruksi dijalankan. Di dalam mikrokontroler *slave* terdapat instruksi pemeriksaan data pendeteksi sayap, yaitu data yang dikirim oleh mikrokontroler *master*, jika data ini bernilai 0, maka prosedur kontrol PID tidak dijalankan, pemeriksaan ini akan terus berlangsung ketika mikrokontroler *master* mengirimkan data.

Pada saat instruksi kontrol PID dijalankan akan terdapat pemeriksaan data *string*, yaitu ’R0600’, jika mikrokontroler *master* mengirimkan data tersebut, maka mikrokontroler *slave* akan menghentikan proses instruksi kontrol PID, pemeriksaan ini akan terus berlangsung selama mikrokontroler *master* mengirim data deteksi sayap bernilai 1.

3.3.3 Diagram Alir Prosedur Pembacaan ADC (pembacaan sensor accelerometer dan tekanan udara)

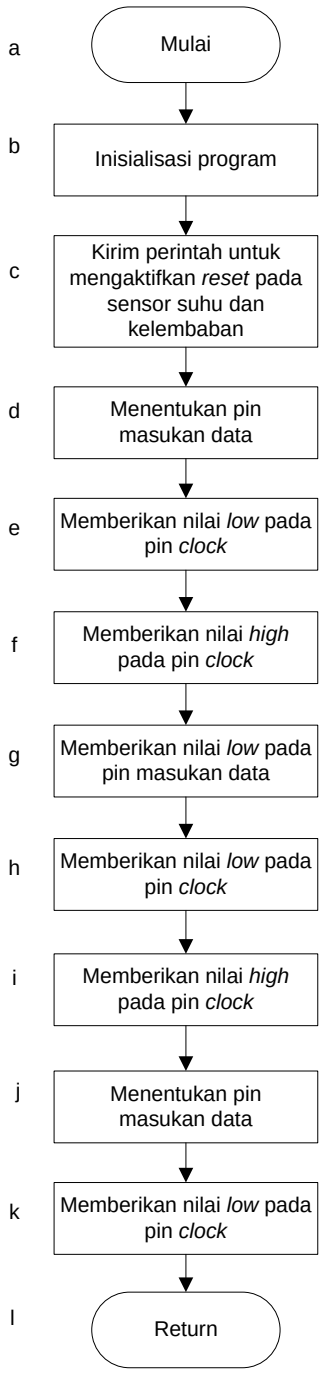


Gambar 3.15 Diagram alir prosedur pembacaan ADC

Berikut ini penjelasan diagram alir prosedur pembacaan ADC:

a. Memulai awal program.
b. Pendeklarasian program. Pengaturan nama prosesor, nama kompiler, konfigurasi pin-pin dan deklarasi variabel.
c. Memberikan nilai *high* pada pin CS ADC.
d. Memberikan nilai *low* pada pin CS ADC.
e. Mengirimkan perintah untuk membaca alamat *channel* yang digunakan untuk pembacaan sensor tekanan udara.
f. Mengirimkan perintah untuk membaca hasil pengukuran tekanan udara dan menyimpannya pada variabel yang disediakan.
g. Hasil pengukuran diubah kedalam satuan *kilopascal* (kPa).
h. Memberikan nilai *high* pada pin CS ADC.
i. Memberikan nilai *low* pada pin CS ADC.
j. Mengirimkan perintah untuk membaca alamat *channel* yang digunakan untuk pembacaan sensor *accelerometer*.
k. Mengirim perintah untuk membaca hasil pengukuran percepatan dan menyimpannya pada variabel yang disediakan.
l. Hasil pengukuran diubah kedalam satuan gravitasi (G).
m. Keluar dari prosedur dan kembali mengeksekusi intruksi berikutnya pada program utama.

3.3.4 Diagram Alir Prosedur Perintah Reset untuk Sensor Suhu dan Kelembaban

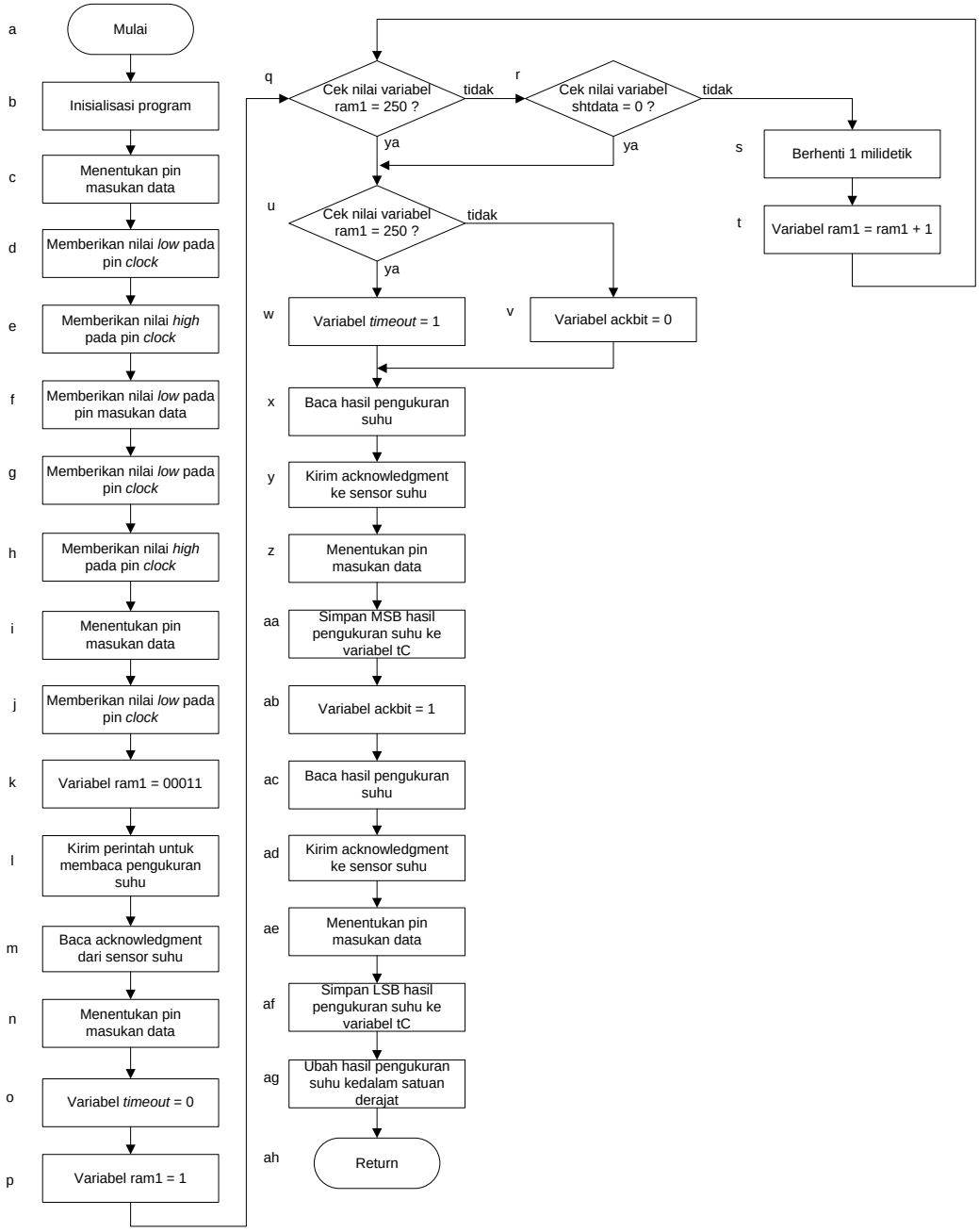


Gambar 3.16 Diagram alir prosedur reset untuk sensor suhu dan kelembaban

Berikut ini penjelasan diagram alir prosedur perintah *reset* untuk sensor suhu dan kelembaban:

a. Memulai awal program.
b. Pendeklarasian program. Pengaturan nama prosesor, nama kompiler, konfigurasi pin-pin dan deklarasi variabel.
c. Pendeklarasian program. Pengaturan nama prosesor, nama kompiler, konfigurasi pin-pin dan deklarasi variabel.
d. Menentukan pin masukan data dari sensor suhu dan kelembaban.
e. Memberikan nilai low pada pin clock sensor suhu dan kelembaban.
f. Memberikan nilai high pada pin clock sensor suhu dan kelembaban.
g. Memberikan nilai low pada pin masukan data dari sensor suhu dan kelembaban.
h. Memberikan nilai low pada pin clock sensor suhu dan kelembaban.
i. Memberikan nilai high pada pin clock sensor suhu dan kelembaban.
j. Menentukan pin masukan data dari sensor suhu dan kelembaban.
k. Memberikan nilai low pada pin clock sensor suhu dan kelembaban.
l. Keluar dari prosedur dan kembali mengeksekusi intruksi berikutnya pada program utama.

3.3.5 Diagram Alir Prosedur Pembacaan Sensor Suhu



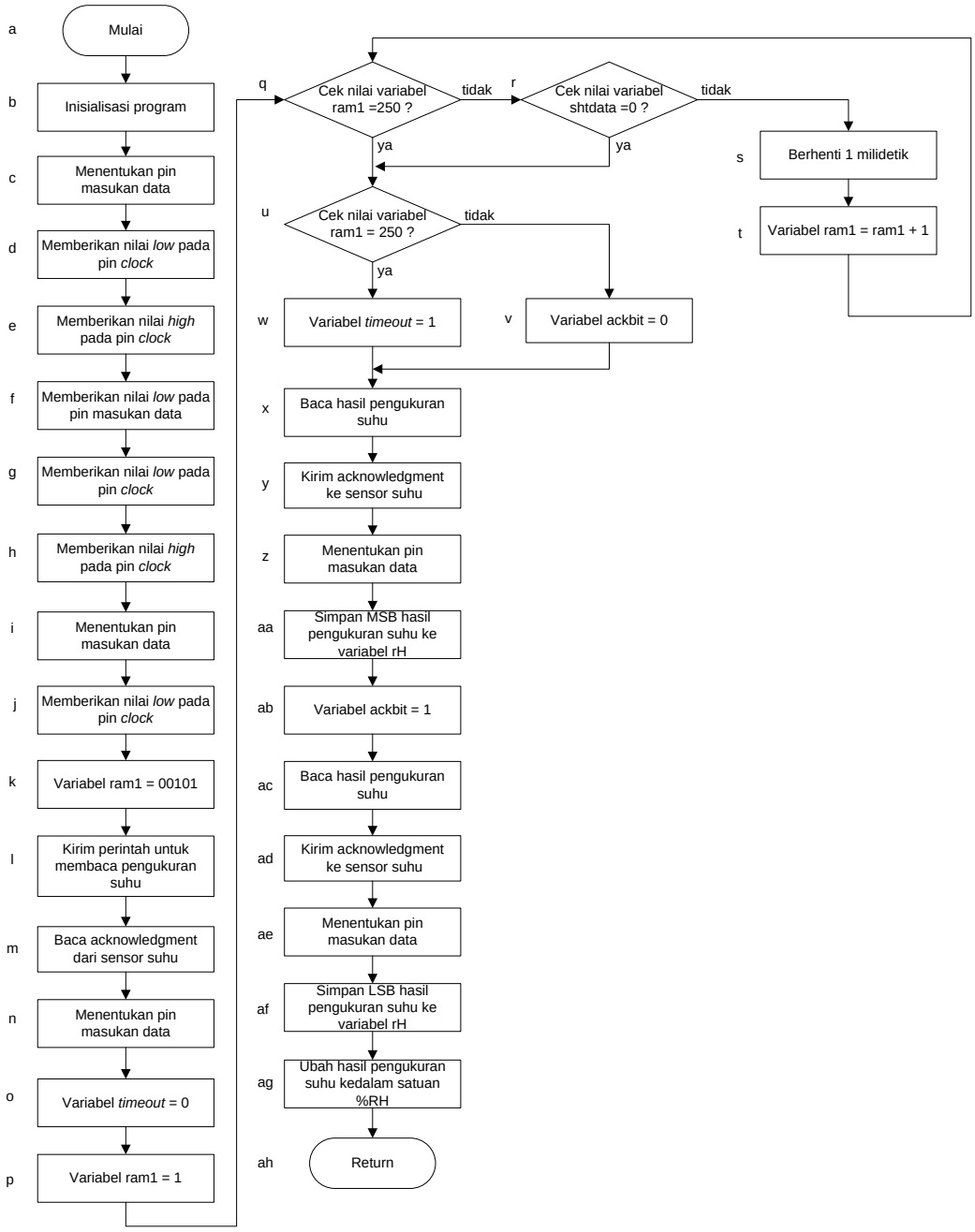
Gambar 3.17 Diagram alir prosedur pembacaan sensor suhu

Berikut ini penjelasan diagram alir prosedur pembacaan sensor suhu:

a. Memulai awal program.
b. Pendeklarasian program. Pengaturan nama prosesor, nama kompiler, konfigurasi pin-pin dan deklarasi variabel.
c. Menentukan pin masukan data dari sensor suhu.
d. Memberikan nilai *low* pada pin *clock* sensor suhu.
e. Memberikan nilai *high* pada pin *clock* sensor suhu.
f. Memberikan nilai *low* pada pin masukan data dari sensor suhu.
g. Memberikan nilai *low* pada pin *clock* sensor suhu.
h. Memberikan nilai *high* pada pin *clock* sensor suhu.
i. Menentukan pin masukan data dari sensor suhu.
j. Memberikan nilai *low* pada pin *clock* sensor suhu.
k. Isi variabel ram1 dengan nilai biner 00011.
l. Mengirim perintah untuk membaca pengukuran suhu.
m. Baca *acknowledgment* dari sensor suhu.
n. Menentukan pin masukan data dari sensor suhu.
o. Isi variabel *timeout* dengan nilai 0.
p. Isi variabel ram1 dengan nilai 1.
q. Pemeriksaan apakah nilai pada variabel ram1 = 250, jika ya lanjutkan ke poin u, jika tidak lanjutkan ke poin r.
r. Pemeriksaan apakah nilai pada variabel shtdata = 0, jika ya lanjutkan ke poin u, jika tidak lanjutkan ke poin s.
s. Berhenti selama 1 milidetik.
t. Jumlahkan nilai pada variabel ram1 dengan nilai 1 dan hasil pejumlahan simpan kembali pada variabel ram1, kemudian lanjutkan ke poin q.
u. Pemeriksaan apakah nilai pada variabel ram1 = 250, jika ya lanjutkan ke poin w, jika tidak lanjutkan ke poin v.
v. Isi variabel ackbit dengan nilai 0.
w. Isi variabel timeout dengan nilai 1.
x. Mengirim perintah untuk membaca hasil data pengukuran suhu.
y. Mengirim perintah *acknowledgment* ke sensor suhu.

z. Menentukan pin masukan data dari sensor suhu.
aa. Menyimpan MSB hasil pengukuran suhu pada variabel tC.
ab. Isi variabel ackbit dengan nilai 1.
ac. Mengirim perintah untuk membaca hasil data pengukuran suhu.
ad. Mengirim perintah *acknowledgment* ke sensor suhu.
ae. Menentukan pin masukan data dari sensor suhu.
af. Menyimpan LSB hasil pengukuran suhu pada variabel tC.
ag. Hasil data pengukuran suhu diubah ke dalam satuan derajat *celcius* (^{o}C).
ah. Keluar dari prosedur dan kembali mengeksekusi instruksi berikutnya pada program utama.

3.3.6 Diagram Alir Prosedur Pembacaan Sensor Kelembaban



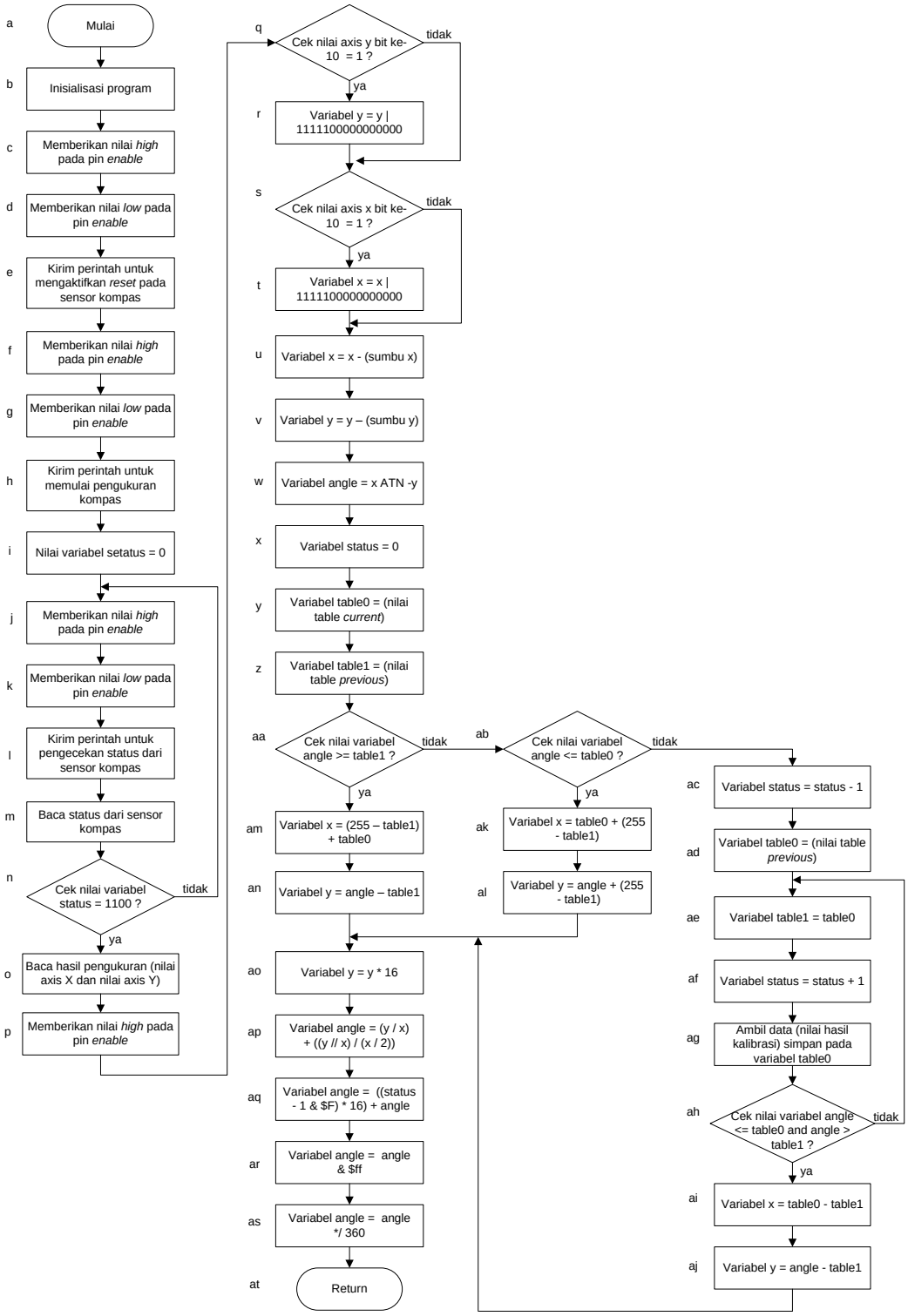
Gambar 3.18 Diagram alir prosedur pembacaan sensor kelembaban

Berikut ini penjelasan diagram alir prosedur pembacaan sensor kelembaban:

a. Memulai awal program.
b. Pendeklarasian program. Pengaturan nama prosesor, nama kompiler, konfigurasi pin-pin dan deklarasi variabel.
c. Menentukan pin masukan data dari sensor kelembaban.
d. Memberikan nilai low pada pin clock sensor kelembaban.
e. Memberikan nilai high pada pin clock sensor kelembaban.
f. Memberikan nilai low pada pin masukan data dari sensor kelembaban.
g. Memberikan nilai low pada pin clock sensor kelembaban.
h. Memberikan nilai high pada pin clock sensor kelembaban.
i. Menentukan pin masukan data dari sensor kelembaban.
j. Memberikan nilai low pada pin clock sensor kelembaban.
k. Isi variabel ram1 dengan nilai biner 00101.
l. Mengirim perintah untuk membaca pengukuran kelembaban.
m. Baca acknowledgment dari sensor kelembaban.
n. Menentukan pin masukan data dari sensor kelembaban.
o. Isi variabel timeout dengan nilai 0.
p. Isi variabel ram1 dengan nilai 1.
q. Pemeriksaan apakah nilai pada variabel ram1 = 250, jika ya lanjutkan ke poin u, jika tidak lanjutkan ke poin r.
r. Pengecekan apakah nilai pada variabel shtdata = 0, jika ya lanjutkan ke poin u, jika tidak lanjutkan ke poin s.
s. Berhenti selama 1 milidetik.
t. Jumlahkan nilai pada variabel ram1 dengan nilai 1 dan hasil penjumlahan simpan kembali pada variabel ram1, kemudian lanjutkan ke poin q.
u. Pemeriksaan apakah nilai pada variabel ram1 = 250, jika ya lanjutkan ke poin w, jika tidak lanjutkan ke poin v.
v. Isi variabel ackbit dengan nilai 0.
w. Isi variabel timeout dengan nilai 1.

x. Mengirim perintah untuk membaca hasil data pengukuran kelembaban.
y. Mengirim perintah acknowledgment ke sensor kelembaban.
z. Menentukan pin masukan data dari sensor kelembaban.
aa. Menyimpan MSB hasil pengukuran kelembaban pada variabel rH.
ab. Isi variabel ackbit dengan nilai 1.
ac. Mengirim perintah untuk membaca hasil data pengukuran kelembaban.
ad. Mengirim perintah acknowledgment ke sensor kelembaban.
ae. Menentukan pin masukan data dari sensor kelembaban.
af. Menyimpan LSB hasil pengukuran kelembaban pada variabel rH.
ag. Hasil data pengukuran kelembaban diubah kedalam satuan %RH.
ah. Keluar dari prosedur dan kembali mengeksekusi instruksi berikutnya pada program utama.

3.3.7 Diagram Alir Prosedur Pembacaan Sensor Kompas



Gambar 3.19 Diagram alir prosedur pembacaan sensor kompas

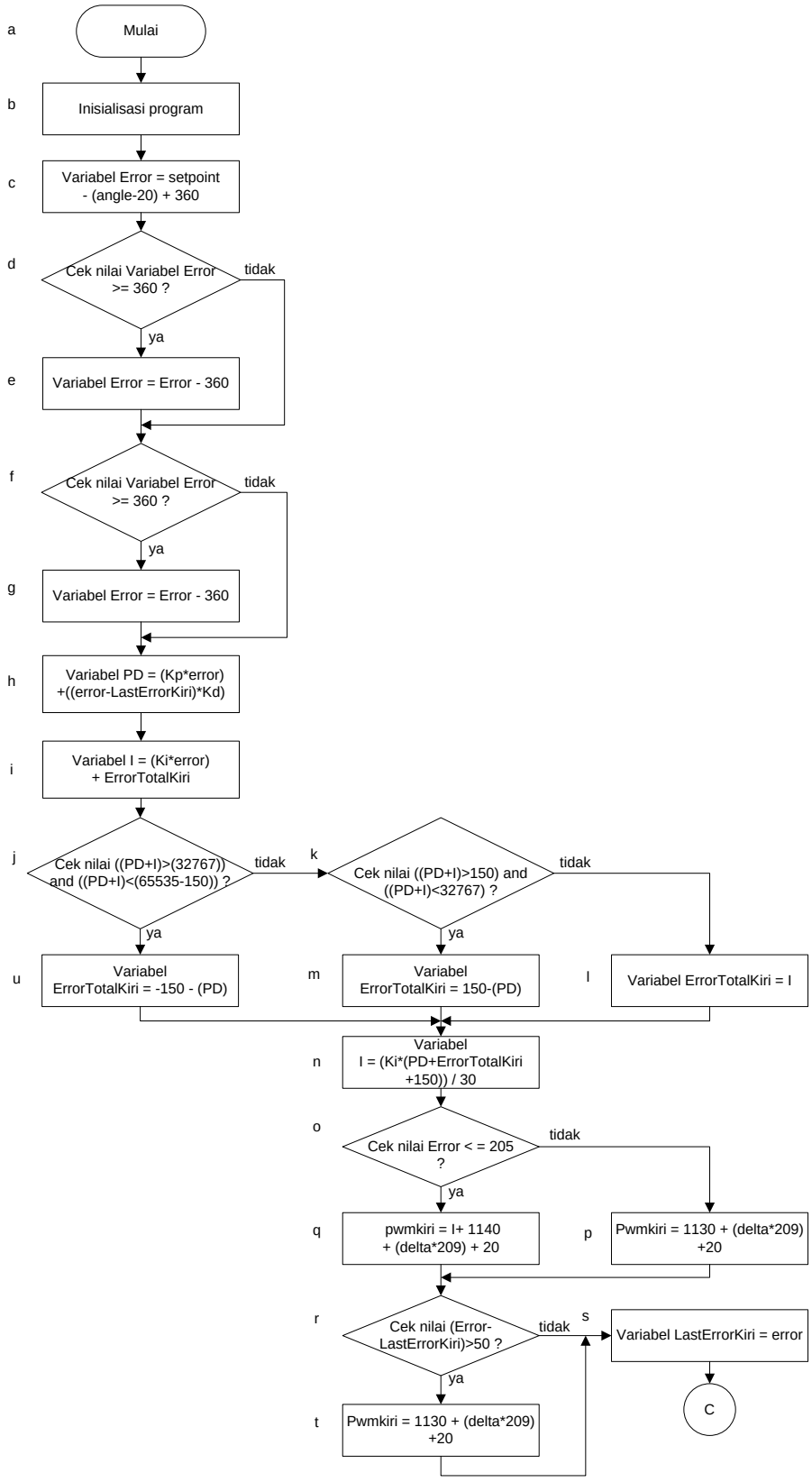
Berikut ini penjelasan diagram alir prosedur pembacaan sensor kompas:

a. Memulai awal program.
b. Pendeklarasian program. Pengaturan nama prosesor, nama kompiler, konfigurasi pin-pin dan deklarasi variabel.
c. Memberikan nilai high pada pin enable sensor kompas.
d. Memberikan nilai low pada pin enable sensor kompas.
e. Kirim perintah untuk mengaktifkan reset pada sensor kompas.
f. Memberikan nilai high pada pin enable sensor kompas.
g. Memberikan nilai low pada pin enable sensor kompas.
h. Kirim perintah untuk memulai pengukuran kompas.
i. Isi variabel status dengan nilai 0.
j. Memberikan nilai high pada pin enable sensor kompas.
k. Memberikan nilai low pada pin enable sensor kompas.
l. Kirim perintah untuk pengecekan status dari sensor kompas.
m. Kirim perintah untuk membaca status dari sensor kompas.
n. Pemeriksaan apakah nilai pada variabel status = 1100, jika ya lanjutkan ke poin o, jika tidak kembali ke poin j.
o. Kirim perintah untuk membaca hasil pengukuran axis X dan axis Y.
p. Memberikan nilai high pada pin enable sensor kompas.
q. Pemeriksaan apakah nilai pada variabel y bit ke-10 = 1, jika ya lanjutkan ke poin r, jika tidak lanjutkan ke poin s.
r. Nilai variabel y di-OR kan dengan nilai 1111100000000000, kemudian hasilnya disimpan kembali pada variabel y.
s. Pemeriksaan apakah nilai pada variabel x bit ke-10 = 1, jika ya lanjutkan ke poin t, jika tidak lanjutkan ke poin u.
t. Nilai variabel x di-OR kan dengan nilai 1111100000000000, kemudian hasilnya disimpan kembali pada variabel x.
u. Kurangi nilai pada variabel x dengan nilai sumbu x, kemudian hasil pengurangan simpan kembali pada variabel x.
v. Kurangi nilai pada variabel y dengan nilai sumbu y, kemudian hasil pengurangan simpan kembali pada variabel y.

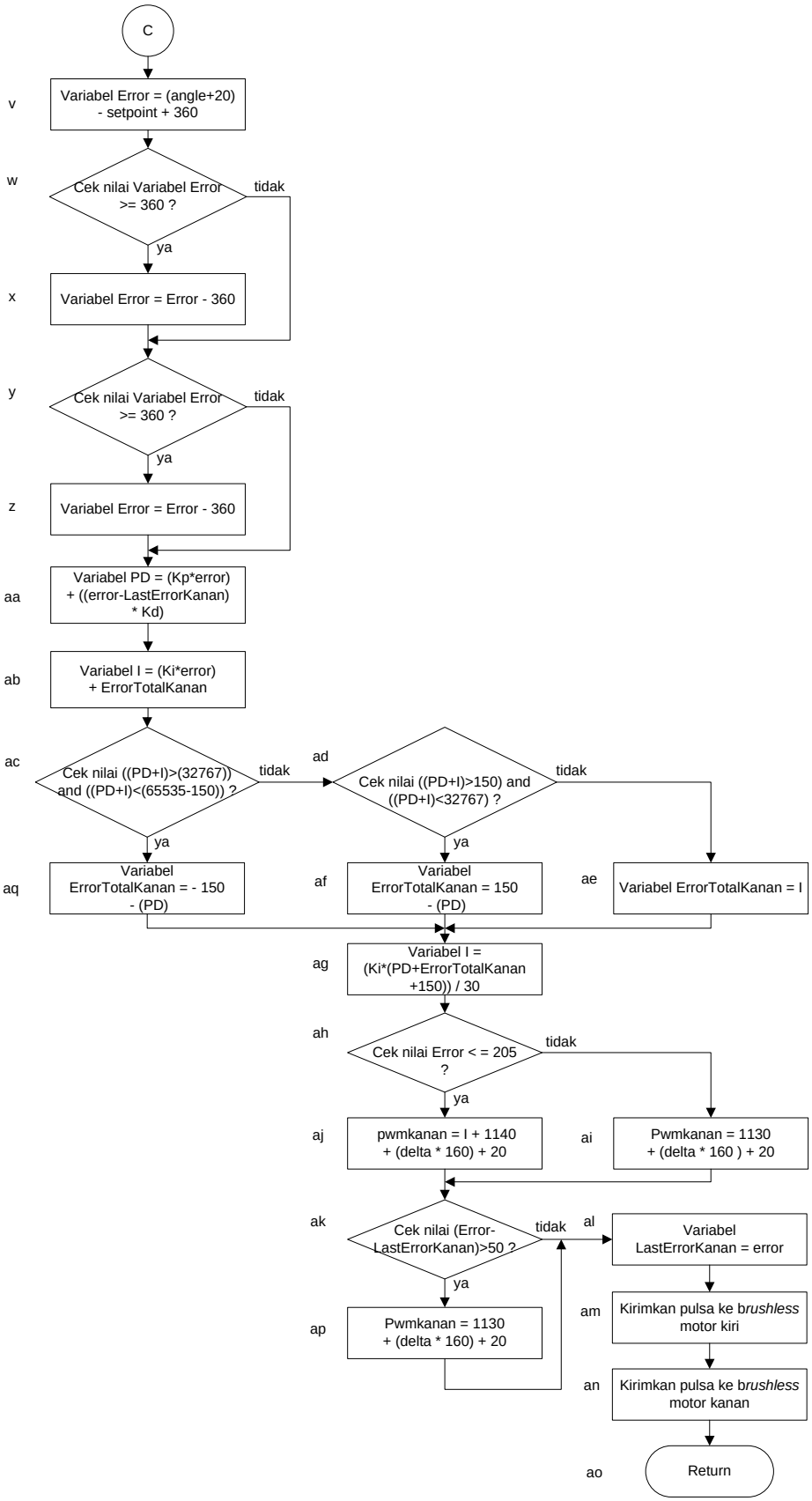
w. Isi variabel angle dengan hasil dari x ATN -y.
x. Isi variabel status dengan nilai 0.
y. Isi variabel table0 dengan nilai table current.
z. Isi variabel table1 dengan nilai table previous.
aa. Pemeriksaan apakah nilai pada variabel angle >= table1, jika ya lanjutkan ke poin am, jika tidak lanjutkan ke poin ab.
ab. Pemeriksaan apakah nilai pada variabel angle < table0, jika ya lanjutkan ke poin ak, jika tidak lanjutkan ke poin ac.
ac. Kurangi nilai pada variabel status dengan nilai 1, kemudian hasil pengurangan simpan kembali pada variabel status.
ad. Isi variabel table0 dengan nilai table previous.
ae. Isi variabel table1 dengan nilai pada variabel table0.
af. Jumlahkan nilai pada variabel status dengan nilai 1, kemudian hasil penjumlahan simpan kembali pada variabel status.
ag. Ambil data (nilai hasil kalibrasi) dan simpan pada table0.
ah. Pemeriksaan apakah nilai pada (variabel angle <= table0) dan (variabel angle > table1), jika ya lanjutkan ke poin ai, jika tidak kembali ke poin ae.
ai. Kurangi nilai pada variabel table0 dengan nilai pada variabel table1, kemudian simpan hasil pengurangan pada variabel x.
aj. Kurangi nilai pada variabel angle dengan nilai pada variabel table1, kemudian hasil pengurangan simpan pada variabel y.
ak. Kurangi nilai 255 dengan nilai pada variabel table1 dan jumlahkan dengan nilai pada variabel table0, kemudian hasilnya simpan pada variabel x.
al. Kurangi nilai 255 dengan nilai pada variabel table1 dan jumlahkan dengan nilai pada variabel angle, kemudian hasilnya simpan pada variabel y.
am. Kurangi nilai 255 dengan nilai pada variabel table1 dan jumlahkan dengan nilai pada variabel table0, kemudian hasilnya simpan pada variabel x.

an. Kurangi nilai pada variabel angle dengan nilai pada variabel table1, kemudian hasil pengurangan simpan pada variabel y.
ao. Kalikan nilai pada variabel y dengan nilai 16, kemudian hasil perkalian simpan kembali pada variabel y.
ap. Hasil dari (y/x)+((y//x)/(x/2)) simpan pada variabel angle.
aq. Hasil dari (((status–1&$f)*16)+angle) simpan pada variabel angle.
ar. Hasil dari (angle & $ff) simpan pada variabel angle.
as. Hasil dari ((angle)*/360) simpan pada variabel angle.
at. Keluar dari prosedur dan kembali mengeksekusi instruksi berikutnya pada program utama.

3.3.8 Diagram Alir Prosedur Kontrol PID



Gambar 3.20 Diagram alir prosedur kontrol PID



Gambar 3.21 Diagram alir prosedur kontrol PID (lanjutan)

Berikut tabel penjelasan diagram alir prosedur kontrol PID:

a. Memulai awal program.
b. Pendeklarasian program. Pengaturan nama prosesor, nama kompiler, konfigurasi pin-pin dan deklarasi variabel.
C. Hasil dari (set point – (angle-20) + 360) simpan pada variabel Error.
d. Pemeriksaan apakah nilai pada variabel Error >= 360, jika ya lanjutkan ke poin e, jika tidak lanjutkan ke poin f.
e. Kurangi nilai pada variabel Error dengan nilai 360, kemudian hasil dari pengurangan simpan kembali pada variabel Error.
f. Pemeriksaan apakah nilai pada variabel Error >= 360, jika ya lanjutkan ke poin g, jika tidak lanjutkan ke poin h.
g. Kurangi nilai pada variabel Error dengan nilai 360, kemudian hasil dari pengurangan simpan kembali pada variabel Error.
h. Hasil dari (Kp*error)+((error-LastErrorKiri)*Kd) simpan pada variabel PD.
i. Hasil dari (Ki*error)+(ErrorTotalKiri) simpan pada variabel I.
j. Pemeriksaan apakah nilai pada variabel ((PD+I) > 32767) dan ((PD+I) < (65535-150)), jika ya lanjutkan ke poin u, jika tidak lanjutkan ke poin k.
k. Pemeriksaan apakah nilai pada variabel ((PD+I) > 150) dan ((PD+I) < 32767), jika ya lanjutkan ke poin m, jika tidak lanjutkan ke poin l.
l. Isi nilai pada variabel ErrorTotalKiri dengan nilai pada variabel I.
m. Kurangi nilai 150 dengan nilai pada variabel PD, kemudian hasil pengurangan simpan pada variabel ErrorTotalKiri.
n. Hasil dari ((Ki*(PD+ErrorTotalKiri+150)) / 30) simpan pada variabel I.
o. Pemeriksaan apakah nilai pada variabel Error <= 205, jika ya lanjutkan ke poin q, jika tidak lanjutkan ke poin p.
p. Hasil dari (1130 + (delta*209)+20) simpan pada variabel Pwmkiri.

q. Hasil dari (I+ 1140 + (delta*209) + 20) simpan pada variabel pwmkiri.
r. Pemeriksaan apakah nilai pada variabel (Error-LastErrorKiri) > 50, jika ya lanjutkan ke poin t, jika tidak lanjutkan ke poin s.
s. Isi nilai pada variabel LastErrorKiri dengan nilaipada variabel Error.
t. Hasil dari ((1130 + (delta*209)+20) simpan pada variabel Pwmkiri.
u. Kurangi nilai (-150) dengan nilai pada variabel PD, kemudian hasil pengurangan simpan pada variabel ErrorTotalKiri.
v. Hasil dari ((angle+20)-set point + 360) simpan pada variabel Error.
w. Pemeriksaan apakah nilai pada variabel Error >= 360, jika ya lanjutkan ke poin x, jika tidak lanjutkan ke poin y.
x. Kurangi nilai pada variabel Error dengan nilai 360, kemudian hasil dari pengurangan simpan kembali pada variabel Error.
y. Pemeriksaan apakah nilai pada variabel Error >= 360, jika ya lanjutkan ke poin z, jika tidak lanjutkan ke poin aa.
z. Kurangi nilai pada variabel Error dengan nilai 360, kemudian hasil dari pengurangan simpan kembali pada variabel Error.
aa. Hasil dari (Kp*error)+((error-LastErrorKiri)*Kd) simpan pada variabel PD.
ab. Hasil dari (Ki**error*)+(ErrorTotalKiri) simpan pada variabel I.
ac. Pemeriksaan apakah nilai pada variabel ((PD+I) > 32767) dan ((PD+I) < (65535-150)), jika ya lanjutkan ke poin aq, jika tidak lanjutkan ke poin ad.
ad. Pemeriksaan apakah nilai pada variabel ((PD+I) > 150) dan ((PD+I) < 32767), jika ya lanjutkan ke poin af, jika tidak lanjutkan ke poin ae.
ae. Isi nilai pada variabel ErrorTotalKanan dengan nilai yang ada pada variabel I.

af. Kurangi nilai 150 dengan nilai pada variabel PD, kemudian hasil pengurangan simpan pada variabel ErrorTotalKanan.
ag. Hasil dari ((Ki*(PD+ErrorTotalKanan+150)) / 30) simpan pada variabel I.
ah. Pemeriksaan apakah nilai pada variabel *Error* <= 205, jika ya lanjutkan ke poin aj, jika tidak lanjutkan ke poin ai.
ai. Hasil dari (1130 + (delta*160)+20) simpan pada variabel Pwmkanan.
aj. Hasil dari (I+ 1140 + (delta*160) + 20) simpan pada variabel pwmkanan.
ak. Pemeriksaan apakah nilai pada variabel (*Error*-LastErrorKanan) > 50, jika ya lanjutkan ke poin ap, jika tidak lanjutkan ke poin al.
al. Isi nilai pada variabel LastErrorKanan dengan nilai pada variabel *Error*.
am. Mengirimkan pulsa ke *brushless* motor kiri sesuai dengan nilai pada variabel pwmkiri.
an. Mengirimkan pulsa ke *brushless* motor kanan sesuai dengan nilai pada variabel pwmkanan.
ao. Keluar dari prosedur dan kembali mengeksekusi intruksi berikutnya pada program utama.
ap. Hasil dari ((1130 + (delta*160)+20) simpan pada variabel Pwmkanan.
aq. Kurangi nilai (-150) dengan nilai pada variabel PD, kemudian hasil pengurangan simpan pada variabel ErrorTotalKanan.

Algoritma di atas diimplementasikan pada bahasa pemograman *basic* melalui *basic stamp editor* agar dapat diisi ke dalam mikrokontroler *basic stamp 2p40.*