BAB III

METODE PENELITIAN

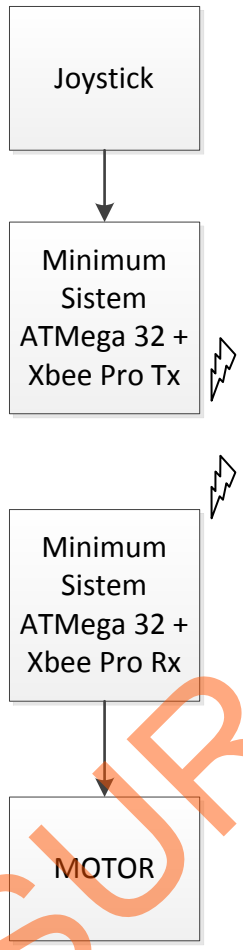
Metode penelitian yang digunakan pada pembuatan perangkat keras dan perangkat lunak yaitu dengan studi kepustakaan. Dengan cara ini penulis berusaha untuk mendapatkan dan mengumpulkan data-data, informasi, konsep-konsep yang bersifat teoritis dari buku bahan-bahan kuliah dan referensi dari internet yang berkaitan dengan permasalahan.

Dari data-data yang diperoleh maka dilakukan perencanaan rangkaian perangkat keras. Dalam perangkat keras ini, penulis akan melakukan pengujian perangkat keras dengan program-program yang telah dibuat, Pembuatan perangkat lunak adalah tahap selanjutnya. Terakhir adalah penggabungan perangkar keras dengan kerja perangkat lunak yang telah selesai dibuat.

Pada bab ini akan dibahas mengenai masalah yang timbul dalam perencanaan dan pembuatan perangkat keras (*hardware*) maupun perangkat lunak (*software*). Dari kedua bagian tersebut akan dipadukan agar dapat bekerja sama untuk menjalankan sistem yang baik.

Perencanaan ini diperlukan sebelum proses pembuatan sistem tersebut, Perancangan ini berguna agar pengerjaan tahapan selanjutnya berjalan dengan lancar. Tahapan-tahapannya meliputi tahap pembuatan perangkat keras, perangkat lunak dan menggabungkan keduanya.

Dalam perancangan perangkat keras kendali PID pada robot manual menggunakan komunikasi nirkabel, penulis menggunakan diagram blok seperti gambar 3.1 dibawah ini :



Gambar 3.1 Blok diagram alat

Dari blok diagram pada gambar 3.1 Kita dapat melihat sistem kendali PID pada robot manual menggunakan komunikasi nirkabel. Pada bagian input, terdapat 2 buah potensio dan 6 tombol *push button* untuk mengatur pergerakan robot. Bagian pemroses yaitu mikrokontroller ATMega32 dibagian *transmiter* & *receiver* bertugas untuk mengirim data dari Xbee-Pro TX untuk dikirimkan ke Xbee-Pro RX dalam bentuk data serial, kemudian dari Xbee-Pro RX data diteruskan ke ATMega 32 dibagian *receiver* untuk memproses data dan menjalankan perintah untuk menjalankan motor. Setelah menjalankan motor,

minimum sistem yang ada pada robot menunggu data kiriman selanjutnya dari minimum pada j*oystick.*

1. Bagian *input Push Button*
    a. *Push button* berfungsi menjalankan perintah untuk menjalankan motor menggerakkan tangan robot naik , turun, ke depan, ke belakang, menjepit & melepas benda.
2. Bagian *input* Potensio
    a. Potensio berfungsi untuk menjalankan perintah melakukan pergerakan motor pada robot maju & mundur
3. Pemroses Mikrokontroller ATmega32
    a. ATmega32 di gunakan pada *transmitter* & *receiver* pada robot manual sebagai transmitter data ke wireless Xbee-Pro TX untuk dikirimkan ke Xbee-Pro RX.
4. *Output*
    a. Mikrokontroller ATMega32 sebagai pengolah data bertugas untuk mengirim data dari Xbee-Pro TX untuk dikirimkan ke Xbee-Pro RX dalam bentuk data serial, kemudian dari Xbee-Pro RX data diteruskan ke ATMega32 dibagian *receiver* untuk memproses data dan menjalankan perintah untuk menjalankan motor. Setelah menjalankan motor, minimum sistem yang ada pada robot menunggu data kiriman selanjutnya dari minimum pada *joystick.*

3.1. Perancangan Perangkat Keras

3.1.1. Perancangan Mekanik Robot

Robot yang digunakan penulis terdiri atas 2 buah roda disertai motor yang terletak disisi kiri dan kanan bagian *base robot* digunakan untuk menjalankan robot, 4 buah motor diletakkan pada tangan robot agar dapat digerakkan naik turun, maju mundur, dan menjepit serta melepas benda. Berikut arsitektur robot secara detail adalah sebagai berikut.

Ukuran dimensi

Ukuran Robot : 500 mm x 600 mm x 1200 mm

Struktur Material

Bahan Material yang digunakan :

a. Bagian Rangka

1. Aluminium Profile
2. Aluminium Sheet.
3. Bearing
4. Katrol
5. Mur dan Baut

b. Bagian dari Penggerak Robot :

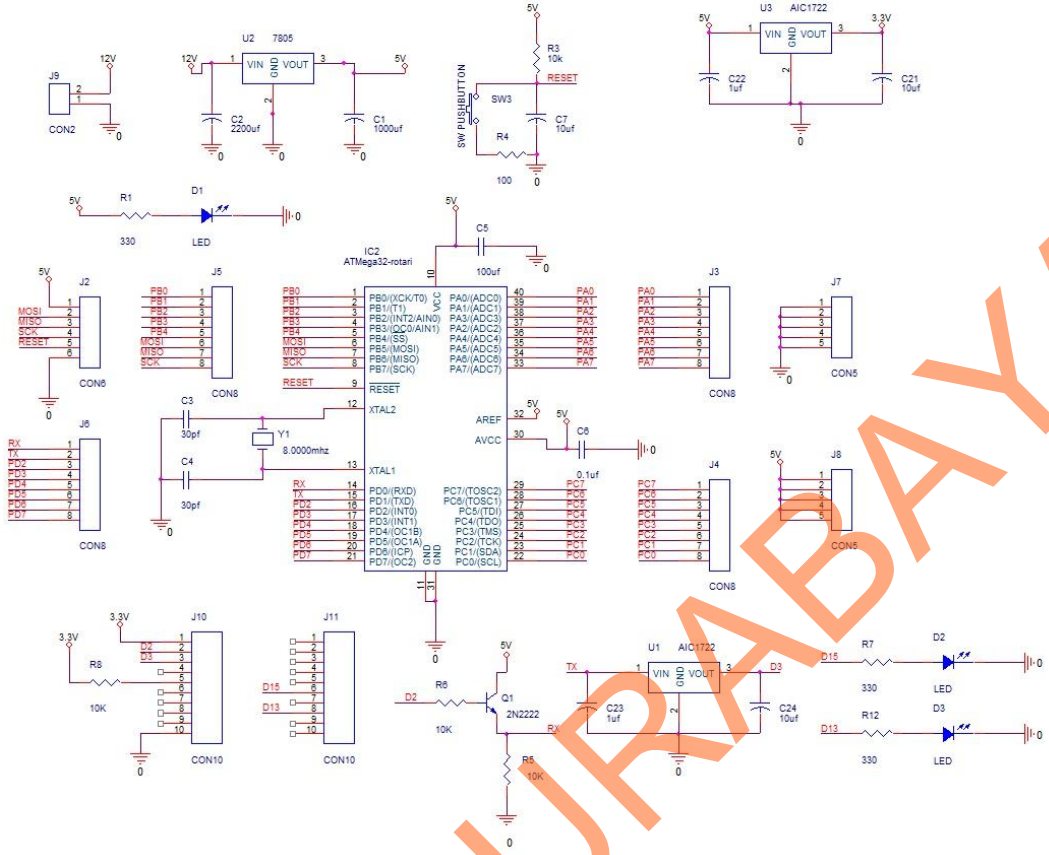
1. Motor DC 24 Volt
2. Aluminium
3. Roda dari karet Silikon
4. Roda Bebas

3.1.2. Perancangan Minimum sistem

Rangkaian minimum sistem dibuat untuk mendukung kerja dari *microchip* ATmega dimana *microchip* tidak bisa berdiri sendiri alias harus ada rangakaian dan komponen pendukung seperti halnya rangakaian catu daya, kristal dan lain sebagaianya yang biasanya disebut minimum sistem .

Microchip berfungsi sebagai otak dalam mengolah semua instruksi baik *input* maupun *output* seperti halnya pemroses data inputan dari potensio dan *push button* kemudian mengirimkan data serial ke Xbee-Pro TX dan memproses data yang diterima dari Xbee-Pro RX kemudian menjalankan perintah untuk menjalankan motor.

Minimum sistem ini dirancang untuk Mikrokontroller ATMega32, dalam perancangannya ini memerlukan beberapa komponen pendukung seperti kristal, resistor dan variabel resistor, dan kapasitor. Rangkaian ini dalam istilah lainnya disebut Minimum sistem ATMega32. *Mikrokontroller* berfungsi untuk memproses data inputan dari potensio dan push button pada *joystick* untuk diteruskan ke Xbee-Pro Tx dan dikirimkan ke Xbee-Pro Rx untuk memerintahkan Minimum sistem receiver mengeksekusi data dengan mengeluarkan output berupa perintah menjalankan motor. Dari proses pengiriman data tersebut, pada Tugas Akhir ini penulis membagi rangkaian minimum sistem menjadi 2(dua) bagian yaitu minimum sistem *transmitter* dan minimum sistem *receiver*. Berikut ini gambar minimum sistem ATMega32 dan rangkaian Xbee-Pro *transmitter* dan *receiver* dapat dilihat seperti Gambar 3.2 dibawah ini.



Gambar 3.2 Minimum sistem ATMega 32 & Rangkaian Xbee-Pro Tx & Rx

Pada rangkaian minimum sistem ATMega 32 *transmitter* dan *receiver* penulis memberikan Pin *VCC* masukan tegangan operasi berkisar antara 4,5 Volt sampai dengan 5 Volt. Pin *RESET* berfungsi untuk masukan *reset* program secara otomatis atau manual. Sedangkan pin MOSI, MISO, dan SCK digunakan untuk keperluan pemrograman *mikrokontroller*. Nilai kapasitor yang digunakan adalah 30 pF. Frekuensi kristal yang dipakai adalah 11,0592 MHz dengan pertimbangan bahwa kristal dengan frekuensi 11,0592 MHz mampu menghasilkan nilai bit TH1, yang digunakan untuk mengatur besarnya nilai *baudrate* yang bulat sehingga *baudrate* yang dihasilkan sama dengan nilai *baudrate* komputer. Untuk melakukan proses *downloading* program dari komputer ke dalam *memory* program internal *mikrokontroller*

3.1.3. Minimum sistem *Transmitter*

Minimum sistem di sisi *transmitter* dirancang untuk mendukung transfer data dari inputan *joystick* yang terdiri dari potensio dan push button pada untuk diteruskan ke Xbee-Pro Tx. Fungsi minimum sistem *trasnmitter* adalah sebagai pembaca data inputan *push button* & potensio. Minimum sistem ini berfungsi untuk mengirimkan data dari inputan *joystick* untuk dikirimkan ke Xbee-Pro *transmitter* agar, dapat diteruskan ke Xbee-Pro *receiver*. Berikut ini adalah blok diagram alur pengiriman data ditunjukkan pada gambar 3.3 dibawah ini.

Gambar 3.3 Blok diagram *minimum sistem transmitter*

Penggunaan Pin pada minimum sistem transmitter adalah sebagai berikut :

a. PortA.0 – PortA.1 digunakan untuk *input* dari potensio.
b. PortB.0 – PortB.5 digunakan untuk *input* dari *push button.*
c. PortD.1 digunakan sebagai *output* untuk mengirimkan data dari potensio dan *push button* untuk diteruskan Xbee-Pro Transmitter.

Rincian penggunaan pin *input* pada mikrokontroller ATMega32 ditunjukkan pada tabel 3.1 & tabel 3.2 dibawah ini.

Tabel.3.1. Pengaturan input pada Mikrokontroller ATMega32 sisi *transmitter*

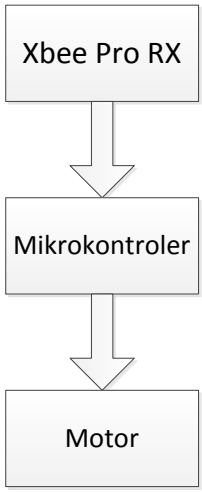
Input	Pin ATMega32
Push button Tangan Naik	PB0
Push button Tangan Turun	PB1
Push button Tangan Maju	PB2
Push button Tangan Mundur	PB3
Push button Tangan Menjepit	PB4
Push button Tangan Melepas	PB5
Potensio Motor *Base* Kanan	PA0
Potensio Motor *Base* Kiri	PA1

Tabel 3.2. Hubungan antara modul pin pada Xbee-Pro & ATMega32

ATMega32		**Xbee-Pro**	
Pin	**Nama**	**Pin**	**Nama**
10	VCC	1	VCC
15	Tx(PD1)	2	Dout (Rx)
14	Rx (PD0)	3	Din (Tx)
11	GND	10	GND

3.1.4. Minimum sistem *Receiver*

Minimum sistem *receiver* berfungsi untuk mengeksekusi perintah yang dikirimkan ke Xbee-pro *receiver* yang telah menerima data kiriman dari minimum sistem *transmitter* kemudian menjalankan perintah untuk menjalankan motor. Untuk perancangan perangkat keras pada minimum sistem *transmittrer* & *receiver* penulis menggunakan blok diagram ditunjukkan pada gambar 3.4 dibawah ini.



Gambar 3.4 Blok diagram *Minimum sistem reciever*

Pada minimum sistem *receiver* ini akan menjalankan eksekusi perintah dengan menjalankan motor dengan mengirimkan perintah pada masing-masing pin pada minimum sistem *receiver* yang difungsikan sebagai *output.* Berikut ini adalah penggunaan pin pada minimum sistem *receiver* ditunjukkan pada tabel 3.3 dan tabel 3.4 dibawah ini.

Tabel.3.3 Pengaturan output pada Mikrokontroller ATMega32 *receiver*

Output	Pin ATMega32
Motor Tangan Naik	PB0
Motor Tangan Turun	PB1
Motor Tangan Maju	PB2
Motor Tangan Mundur	PB3
Motor Tangan Menjepit	PB4
Motor Tangan Melepas	PB5
Motor *Base* Kanan	PA0
Motor *Base* Kiri	PA1

Tabel 3.4 Hubungan antara modul pin pada Xbee-Pro & ATMega32

ATMega32		**Xbee-Pro**	
Pin	**Nama**	**Pin**	**Nama**
10	VCC	1	VCC
15	Tx(PD1)	2	Dout (Rx)
14	Rx (PD0)	3	Din (Tx)
11	GND	10	GND

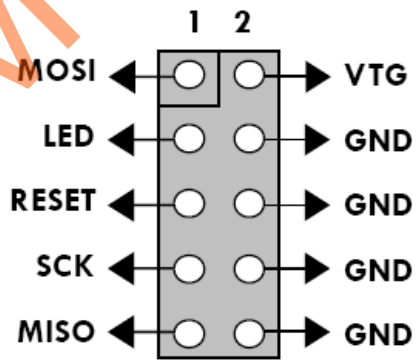
3.1.5. *Downloader*

Untuk melakukan proses *download* program, yaitu file dengan ekstensi ".hex" digunakan perangkat bantu AVR USB ISP yang akan dihubungkan dengan *port* USB (*Universal Serial Bus*) pada komputer. Sebelum *downloader* dapat digunakan perlu dilakukan instalasi *driver* terlebih dahulu. Konfigurasi *pinout* dan keterangan dari *downloader* terdapat pada Tabel 3.5 dan Gambar 3.5.

Tabel 3.5 Keterangan *pinout* AVR USB ISP

Nama	No. Pin	I/O	Keterangan
VTG	2	-	Catu daya dari target *board* (2.7 V - 5.5 V)
GND	4, 6, 8, 10	-	Titik referensi
LED	3	Output	Sinyal kontrol untuk LED (*Light Emitting Diode*) atau *multiplexer* (*optional*)
MOSI	1	Output	*Command* dan data dari AVR USB ISP ke target AVR
MISO	9	Input	Data dari target AVR ke AVR USB ISP
SCK	7	Output	Serial *Clock*, dikendalikan oleh AVR USB ISP
RESET	5	Output	*Reset*, dikendalikan oleh AVR USB ISP

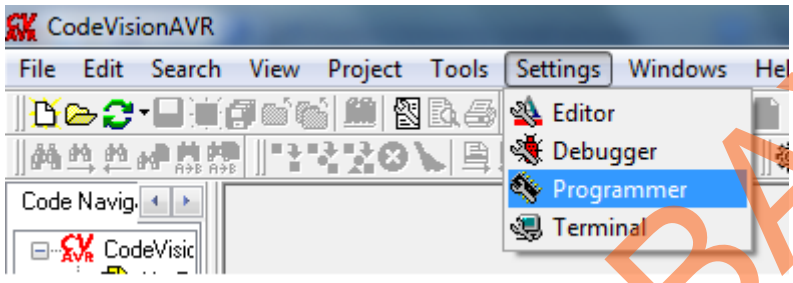
Sumber: INNOVATIVE ELECTRONICS (2009)



Gambar 3.5 *Pinout* AVR USB ISP (INNOVATIVE ELECTRONICS, 2009)

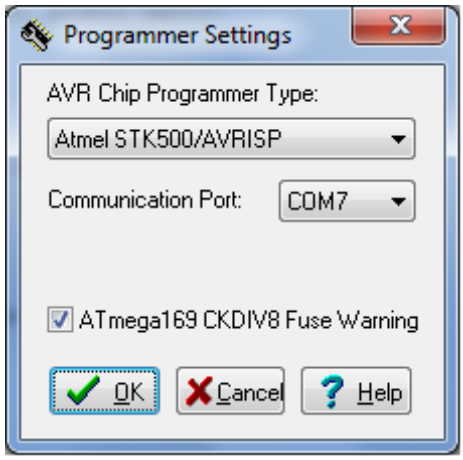
Pin MOSI, pin MISO, pin SCK, pin RESET, dan pin VTG pada AVR USB ISP masing-masing akan dihubungkan pada pin MOSI, pin MISO, pin SCK, pin RESET, dan pin VCC pada *mikrokontroller*. Program editor dan *compiler* yang digunakan untuk pembuatan program adalah *Code Vision* AVR. Proses

download file ".hex" dapat dilakukan melalui program ini. Pengaturan penggunaan *downloader* pada *Code Vision* AVR dilakukan dengan memilih menu *Setting*, kemudian pilihan *Programmer* seperti yang ditunjukkan pada Gambar 3.6.

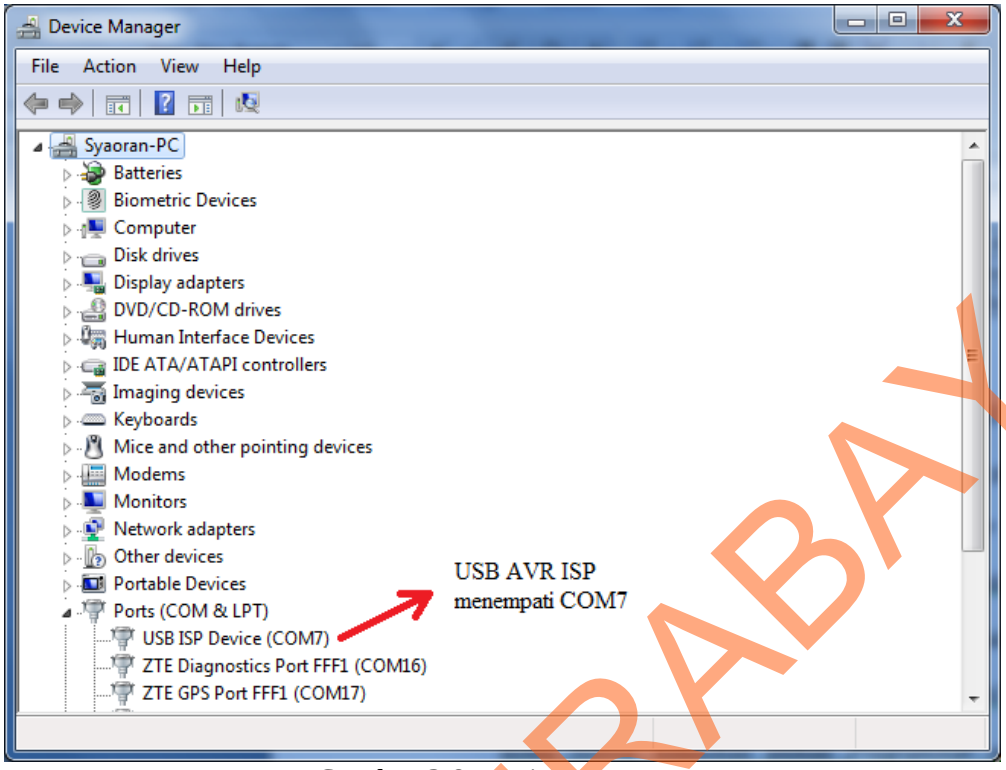


Gambar 3.6 Pemilihan *Programmer* pada menu *Setting* di *Code Vision* AVR

Setelah memilih *Programmer* pada menu *Setting*, akan muncul *window Programmer Setting* seperti pada Gambar 3.6, yang dilanjutkan dengan memilih tipe *programmer* AVR yaitu Atmel STK500/AVRISP. Pilihan *Communication Port* disesuaikan dengan nilai COM yang digunakan oleh *downloader*. Nilai COM dari *downloader* dapat ditemukan pada *Device Manager* bagian *Ports* seperti pada Gambar 3.7.



Gambar 3.7 *Window Programmer Setting* pada *Code Vision* AVR



Gambar 3.8 *Device Manager*

3.1.6. Kontroler PID (*Proportional-Integral-Derivative*)

Guna memperhalus, meningkatkan performansi kontroler dan mendapatkan respon sistem yang baik pada motor. Pada tugas akhir ini penulis menggunakan Kontroler PID (*Proportional-Integral-Derivative*) dengan cara mengimplementasikan pengontrol PID (*Proportional-Integral-Derivative*) pada mikrokontroler ATMega 32 untuk pemroses algoritma PID. Dibawah ini merupakan dari formula PID yang sebelumnya dibahas pada landasan teori PID pada BAB II untuk kemudian dijadikan algoritma PID adalah sebagai berikut :

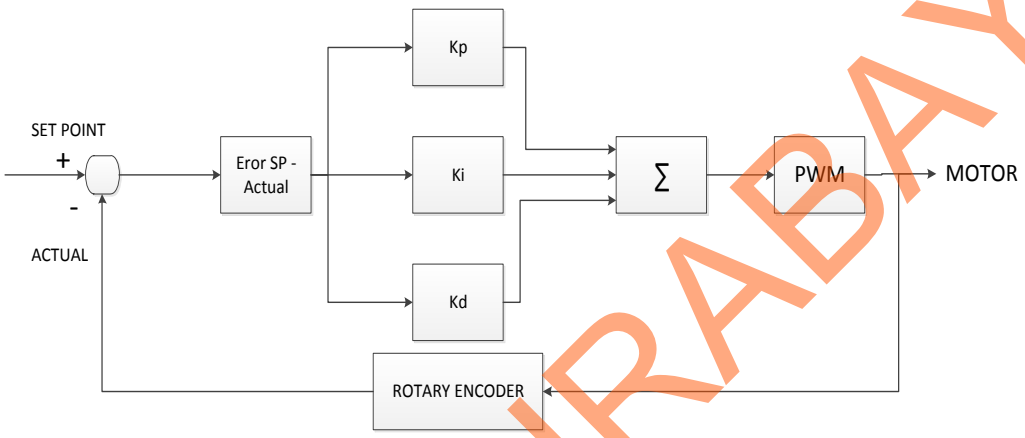
```
r_mot  = r_old + Kp*(e_func-e_old) + Ki*(e_func+e_old)/2
                + Kd*(e_func - 2* e_old + e_old2);
```

Dimana,

r_mot : Hasil penghitungan PID
r_old : Hasil penghitungan PID sebelumnya
Kp : konstanta P

e_func : error sekarang
e_old : error sebelumnya
e_old2 : error dua kali sebelumnya
Ki : konstanta I
Kd : konstanta D

Diagram blok sistem kontrol PID yang akan diimplementasikan pada tugas akhir ini dapat dilihat pada Gambar 3.9.



Gambar 3.9 Diagram blok implementasi PID

Secara umum fungsi dari masing-masing kontroler dalam kontroler PID adalah sebagai berikut :

- Proporsional
  - Berfungsi untuk mempercepat terjadinya respons terhadap sinyal error.
  - Bekerja efektif pada daerah sebelum sistem mencapai daerah setpoint/kondisi start.
- Integral
  - Berfungsi memlihara sinyal kontrol konstan.
  - Bekerja efektif pada daerah di mana sistem mencapai set point.
- Derivatif
  - Berfungsi mendapatkan sinyal kontrol dari perubahan errornya.
  - Bekerja efektif pada daerah transient.

Dari gambar 3.9 terlihat bahwa sistem kontrol motor bekerja dengan mengumpanbalikkan kecepatan motor aktual dan membandingkan dengan kecepatan motor yang diinginkan yaitu set point berdasarkan *error* antara kecepatan motor aktual dengan kecepatan yang diinginkan oleh *user* yang diinputkan melalui potensio yang merupakan set point (SP). Setelah mendapat set point dengan keluaran berupa RPM yang didapatkan dari inputan potensio dan akan membaca aktual dari inputan yang telah dipasang di motor dc yaitu berupa rotary encoder yang akan menghasilkan keluaran RPM, kemudian mikrokontroler ATMega 32 akan melakukan perhitungan algoritma PID untuk menghasilkan output yang dapat meminimalisir *error* sehingga kecepatan motor aktual akan selalu dapat mengikuti kecepatan motor yang diinginkan. Kecepatan motor aktual disebut sebagai variabel proses, sementara kecepatan motor yang diinginkan disebut sebagai set point. Setelah dilakukan perhitungan algoritma PID oleh mikrokontroler ATMega 32, kemudian hasil perhitungan akan dijumlahkan untuk selanjutnya menset PWM motor. Untuk menjembatani keluaran analog dari potensio dengan mikrokontroler ATMega 32 sisi penerima sebelumnya digunakan ADC (Analog Digital to Converter) untuk mengubah sinyal analog menjadi sinyal digital.

3.1.7. Xbee-Pro TX & RX

XBee merupakan suatu modul yang didesain untuk memenuhi standar zigbee/ IEEE 802.15.4 yang biasa digunakan untuk aplikasi jaringan sensor yang berbiaya dan berdaya rendah. Modul ini membutuhkan daya minimal dan menyediakan transfer data yang handal antara dua *device*. Modul ini mempunyai

dimensi fisik kecil sehingga praktis dalam penempatan. Modul ini beroperasi pada rentang frekuensi 2.4 GHz.

3.1.7. Driver Modul Xbee-Pro

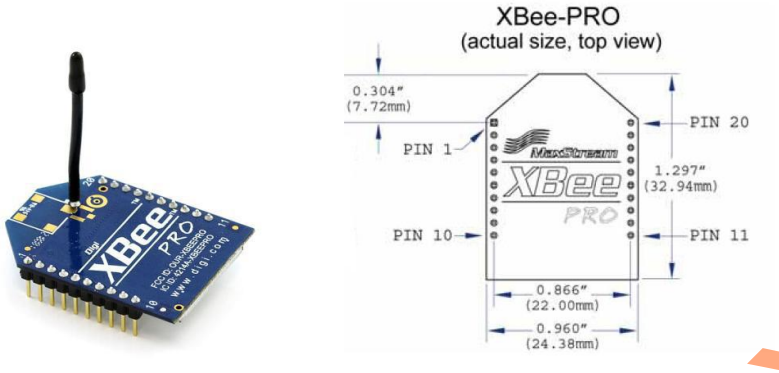
Untuk mengakses modul Xbee-Pro, diperlukan sebuah driver untuk modul Xbee-Pro yang mana modul Xbee-Pro ini hanya memiliki tegangan catu daya rendah yaitu antara 2,8 Volt sampai 3.4 Volt, sehingga diperlukan regulator tegangan sebesar 3.3 Volt, namun untuk data *interface* dapat dihubungkan secara langsung ke mikrokontroller. Berikut adalah hubungan antara modul pin Xbee-Pro dan ATMega32 ditunjukkan pada tabel 3.6 dibawah ini :

Tabel 3.6. Hubungan antara modul pin pada Xbee-Pro & ATMega32

ATMega32		**Xbee-Pro**	
Pin	**Nama**	**Pin**	**Nama**
10	VCC	1	VCC
15	Tx(PD1)	2	Dout (Rx)
14	Rx (PD0)	3	Din (Tx)
11	GND	10	GND

3.1.9. Konfigurasi Pin Xbee-Pro

Berikut adalah gambar & konfigurasi pin Xbee-Pro dan penjelasannya, ditunjukkan pada gambar 3.10 dibawah ini, sedangkan untuk penjelasan fungsi-fungsi setiap pin pada Xbee-Pro dapat dilihat pada tabel 3.7 & untuk spesifikasi Xbee-Pro dapat dilihat pada tabel 3.8.



Gambar 3.10 Modul Xbee-Pro & Dimensi Xbee-Pro

Tabel 3.7 Konfigurasi pin Xbee-Pro

Pin #	Name	Direction	Description
1	VCC	-	Power supply
2	DOUT	Output	UART Data Out
3	DIN / $\overline{\text{CONFIG}}$	Input	UART Data In
4	CD / DOUT_EN / DO8	Output	Carrier Detect, TX_enable or Digital Output 8
5	$\overline{\text{RESET}}$	Input	Module Reset
6	PWM0 / RSSI	Output	PWM Output 0 or RX Signal Strength Indicator
7	[reserved]	-	Do not connect
8	[reserved]	-	Do not connect
9	DTR / SLEEP_RQ / DI8	Input	Pin Sleep Control Line or Digital Input 8
10	GND	-	Ground
11	AD4 / DIO4 / RF_TX	Either	Analog Input 4, Digital I/O 4 or Transmission Indicator
12	DIO7 / $\overline{\text{CTS}}$	Either	Digital I/O 7 or Clear-to-Send Flow Control
13	ON / $\overline{\text{SLEEP}}$	Output	Module Status Indicator
14	VREF	Input	Voltage Reference for A/D Inputs
15	AD5 / DIO5 / Associate	Either	Analog Input 5, Digital I/O 5 or Associated Indicator
16	AD6 / DIO6 / $\overline{\text{RTS}}$	Either	Analog Input 6, Digital I/O 6 or Request-to-Send Flow Control
17	AD3 / DIO3 / COORD_SEL	Either	Analog Input 3, Digital I/O 3 or Coordinator Select
18	AD2 / DIO2	Either	Analog Input 2 or Digital I/O 2
19	AD1 / DIO1	Either	Analog Input 1 or Digital I/O 1
20	AD0 / DIO0	Either	Analog Input 0 or Digital I/O 0

Tabel 3.8 Spesifikasi Xbee-Pro

Specification	XBee-PRO
Performance	
Indoor/Urban Range	up to 300 ft. (100 m)
Outdoor RF line-of-sight Range	up to 4000 ft. (1200 m)
Transmit Power Output	60 mW (18 dBm), 100 mW EIRP
RF Data Rate	250,000 bps
Receiver Sensitivity	-100 dBm (1% PER)
Power Requirements	
Supply Voltage	2.8 – 3.4 V
Transmit Current (typical)	270 mA (@ 3.3 V)
Receive Current (typical)	55 mA (@ 3.3 V)
Power-down Current	< 10 µA
General	
Operating Frequency	ISM 2.4 GHz
Dimensions	0.960" x 1.297" (2.438cm x 3.294cm)
Operating Temperature	-40 to 85° C (industrial)
Antenna Options	U.FL Connector, Chip Antenna or Wire Antenna
Networking & Security	
Supported Network Topologies	Point-to-Point, Point-to-Multipoint, Peer-to-Peer and Mesh
Number of Channels	16 Direct Sequence Channels (software selectable)
Network Layers	PAN ID and Source/Destination Addressing
Agency Approvals	
FCC Part 15.247	pending
Industry Canada (IC)	pending
Europe	pending

3.1.10. *Joystick*

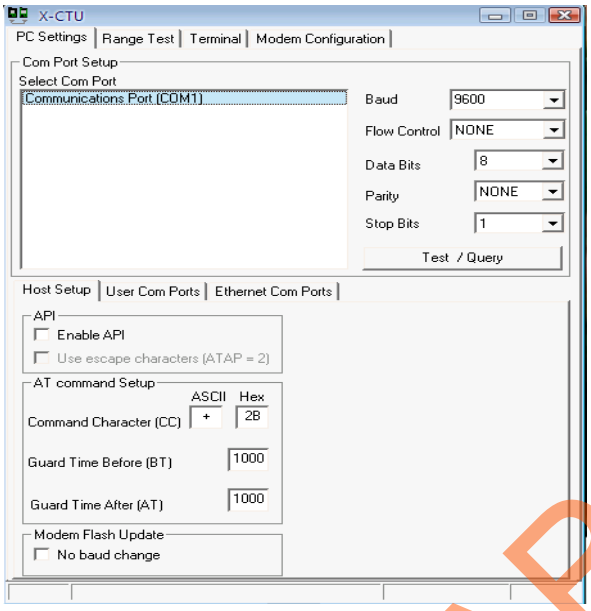
Joystick ini digunakan untuk mengontrol, mengendalikan dan menjalankan robot. Dalam joystick ini terdapat enam buah *push button* dan dua buah potensio yang terdiri dari 2 buah push button yang berfungsi untuk menggerakan tangan robot naik dan turun, dua buah push button yang berfungsi untuk menggerakkan tangan robot maju dan mundur, 2 buah push button yang berfungsi untuk menggerakkan tangan robot untuk menjepit dan melepaskan benda, serta 2 buah potensio yang berfungsi untuk mengontrol motor *base* kanan dan motor *base* kiri yang berguna untuk menjalankan robot.

3.2. Perancangan Perangkat Lunak

Perancangan perangkat lunak bertujuan untuk mengirimkan data dari inputan *user* untuk dieksekusi mikrokontroler untuk menjalankan motor, juga digunakan untuk mendeteksi *error* yang diperoleh dari kontroler PID. Perangkat lunak terbagi dalam beberapa *device* sistem antara lain : konfigurasi Xbee-Pro Tx & Rx, program pengiriman data antar mikrokontroler dari sisi *transmitter* & *receiver*, program LCD *display*, dan program kontroler PID.

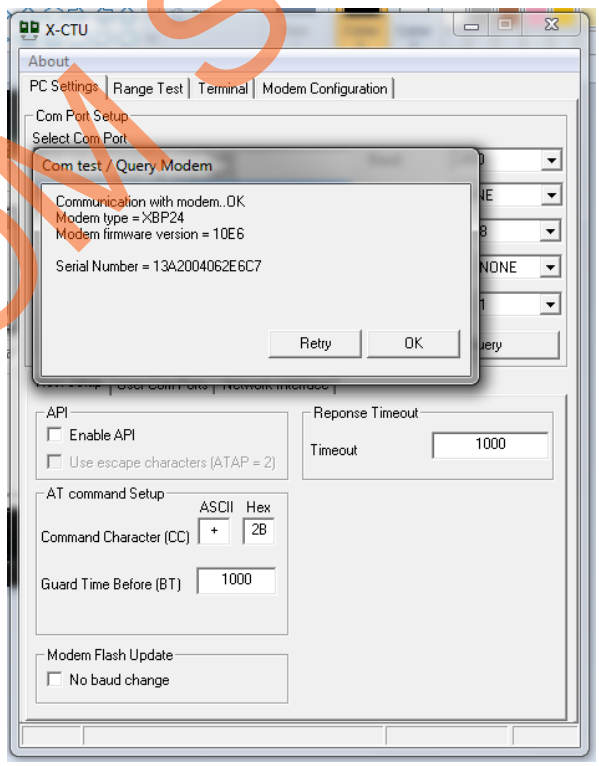
3.2.1. Konfigurasi Parameter Xbee-Pro Tx & Rx

Untuk melakukan konfigurasi parameter modul Xbee-Pro dapat melalui Hyperterminal Windows dan sebuah *software* bawaan Xbee-PRO yaitu X-CTU. Agar Xbee- PRO dapat melakukan komunikasi point to point atau point to multipoint adalah dengan melakukan setting konfigurasi alamat (address). Untuk masuk ke mode konfigurasi pada XCTU, Xbee-PRO harus dihubungkan dengan komputer melalui serial port menggunakan kabel DB9. Setelah terhubung dan *communications port* muncul, buka software XCTU yang merupakan bawaan dari XBee-PRO dengan terlebih dahulu mengatur *baudrate default* sebesar 9600, kemudian klik Test/Query. Seperti ditunjukkan pada gambar 3.11 dibawah ini.



Gambar 3.11 Tampilan untuk setting konfigurasi parameter pada X-CTU

Jika Xbee-PRO berhasil terhubung dengan *software* X-CTU, maka akan keluar jendela baru yang menunjukkan keterangan tipe, *firmware* & *Serial Number* Xbee-PRO. Seperti ditunjukkan pada gambar 3.12 dibawah ini



Gambar 3.12 Informasi Xbee-Pro setelah berhasil terhubung dengan XCTU

Setelah Xbee-Pro terhubung dengan *software* XCTU, dilanjutkan melakukan setting konfigurasi parameter Xbee-Pro di terminal XCTU agar Xbee-Pro di sisi Tx & Rx dapat berkomunikasi dengan baik. Terdapat beberapa perintah untuk mensetting Xbee-Pro. Berikut perintah yang diperlukan untuk mensetting parameter Xbee-Pro dengan menggunakan software X-CTU adalah sebagai berikut.

1 "+++" merupakan perintah untuk memastikan Xbee-Pro siap disetting atau tidak dan juga untuk mengawali setting parameter pada Xbee-Pro.

2 "AT" (AT Command) Merupakan perintah awalan penulisan perintah pada Xbee-Pro.

3 "DL" (Destination Address Low) Merupakan perintah untuk mensetting alamat yang akan dituju oleh Xbee-Pro.

4 "MY" (Source Address) Merupakan perintah untuk mensetting alamat dari Xbee-Pro (alamat diri sendiri), nilai dari "DL" dan "MY" tidak boleh sama.

5 "CH" (Chanel) Merupakan perintah set/read dari Xbee-Pro dimana nilai awal settingnya adalah C dan nilainya harus sama untuk Rx dan Tx.

6 "ID" (Networking {Addressing}) Merupakan perintah pengalamatan PAN (Personal Area Network) dimana nilainya harus sama untuk satu jaringan.

7 "WR"(Write) Merupakan perintah penulisan pada Xbee-Pro, apakah Xbee-Pro siap untuk mengirimkan data.

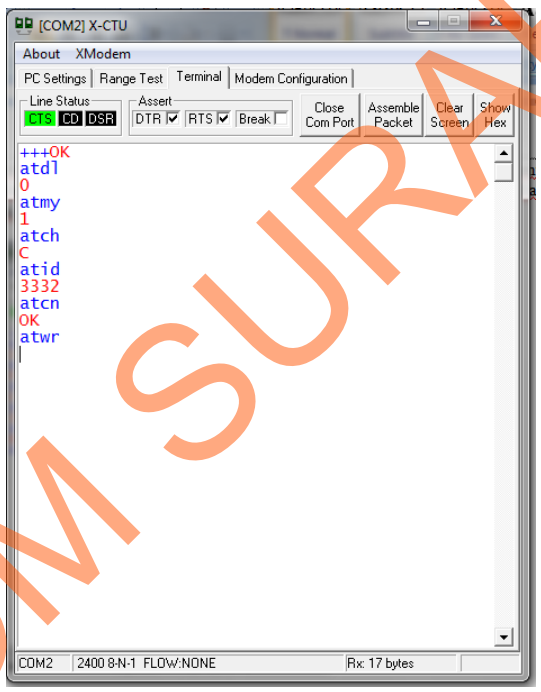
8 "CN" (Exit Command Mode) merupakan perintah keluar dari ATCommand

Agar 2 buah XBee-Pro dapat saling berkomunikasi, maka XBee tersebut harus :

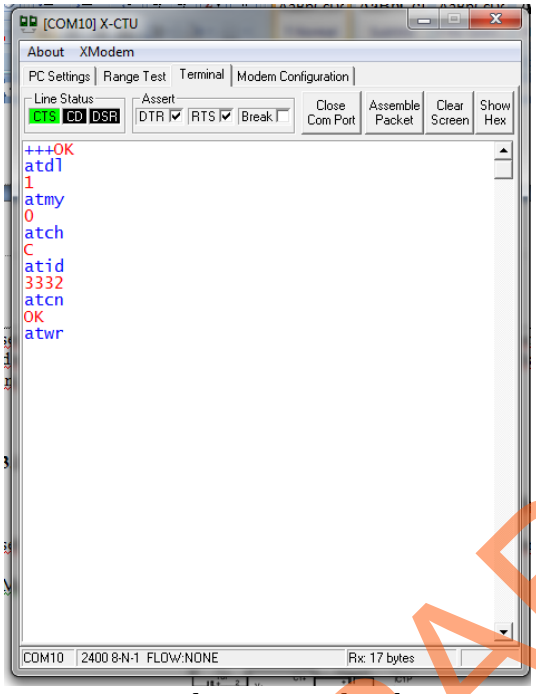
- Mempunyai channel ID (CH) yang sama.

- Mempunyai network ID PAN ID) yang sama.
- Source ID XBee-Pro receiver harus sesuai dengan destination ID dari XBee-Pro *transmitter*

Setting konfigurasi parameter yang harus dilakukan untuk menghubungkan Xbee-Pro berkomunikasi *point to point* yaitu dengan mengetikkan perintah pada Terminal XCTU ditunjukkan pada gambar 3.13 dan gambar 3.14 dibawah ini.



Gambar 3.13 Parameter yang disetting pada Xbee-Pro sisi *Transmitter* (Tx)



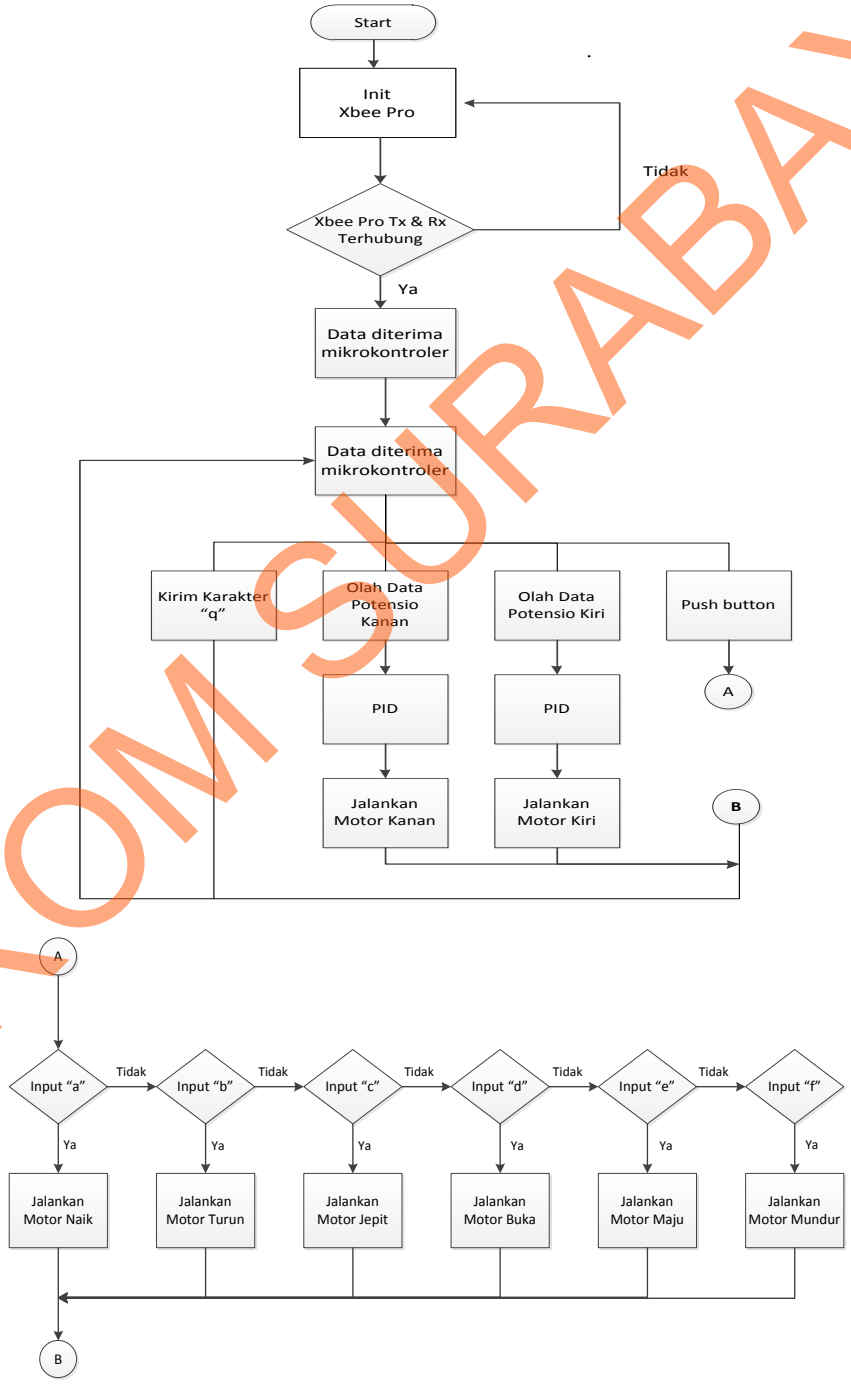
Gambar 3.14 Parameter yang disetting pada Xbee-Pro sisi Receiver (Rx)

Setelah semua setting parameter selesai dilakukan, kemudian mensetting *baudrate* menjadi 2400. Pada tugas akhir ini, penulis menggunakan *baudrate* rendah yaitu 2400, karena kebutuhan pengiriman data yang besar dan untuk mencegah *overflow* (hilangnya data antara *host* dan modul Xbee-Pro). Sebelumnya telah dicoba settingan *baudrate* standart dari Xbee-Pro menggunakan *baudrate* 9600 dan sering terjadi *overflow.*

Pada pengoperasian Xbee-Pro yang digunakan pada tugas akhir ini, menggunakan Mode *transparent* (AT) yang bertujuan untuk mengkonfigurasi *point-to-point* sederhana., yaitu Xbee-Pro bertindak sebagai modem serial antara mikrokontroler dengan *joystick.*

3.2.2. Program Mikrokontroler

A. Diagram alir untuk menampilkan nilai inputan potensio & *push button* pada LCD *Display* & penerimaan data dari minimum sistem Tx dari *user* terdapat pada gambar 3.15 dibawah ini.



Gambar 3.15. Diagram alir program penerimaan data dari minimum sistem Tx

Berikut potongan program LCD *Display* & program penerimaan data dari minimum sistem Tx

```
    // Alphanumeric LCD initialization
    // Connections specified in the
    //  Project|Configure|C  Compiler|Libraries|Alphanumeric  LCD
menu:
    // RS - PORTC Bit 0
    // RD - PORTC Bit 1
    // EN - PORTC Bit 2
    // D4 - PORTC Bit 4
    // D5 - PORTC Bit 5
    // D6 - PORTC Bit 6
    // D7 - PORTC Bit 7
    // Characters/line: 16
    lcd_init(16);

    // Global enable interrupts
    #asm("sei")
    lcd_gotoxy(0,0);
    lcd_putsf("== coba_pwm ==");
    delay_ms(1000);
    lcd_clear();

    //naik = turun =  kanan = kiri =1;
    percepatan = 700 / 115;
    PORTB.0 = PORTB.1 = PORTB.2 = PORTB.3 = PORTB.4 = PORTB.5
=0;
    /*
    ratusan = 0,puluhan = 0,satuan  = 0 ,temp = 0;
    ratusan1 = 0,puluhan1 = 0,satuan1  = 0 ,temp1 = 0; */

    while (1)
          {
          // Place your code here

             while (rx_counter < 3)
             putchar('q');
             potensio_kanan = (int)getchar();
             potensio_kiri = (int)getchar();
             potensiokanan();
             input = getchar();
             tombol();
             //putchar('q');

          }
    }

    void tombol()
    {
        //while (rx_counter < 2);
        //input = getchar();
            if(input == 'a')
            {
                    PORTB.0 = 1;
                    delay_ms(100);
                    input = 0;
```

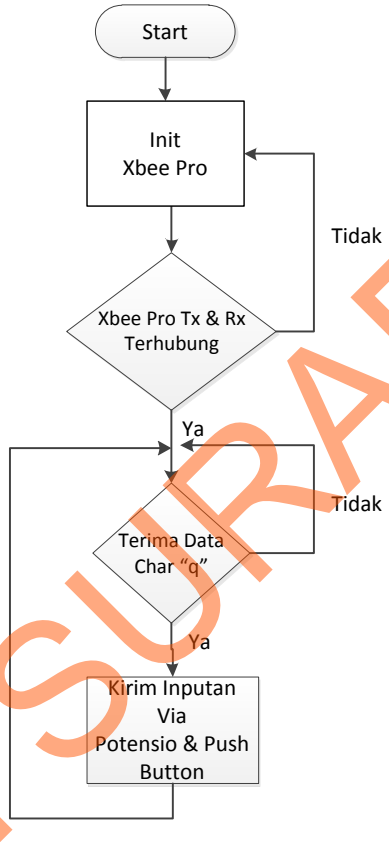
```
                lcd_gotoxy(15,0);
                lcd_putsf("a");
                lcd_putsf(" ");
        }
        else if(input == 'b')
        {
                PORTB.1 = 1;
                delay_ms(100);
                input = 0;
                lcd_gotoxy(15,0);
                lcd_putsf("b");
                lcd_putsf(" ");
        }
        else if(input == 'c')
        {
                PORTB.2 = 1;
                delay_ms(100);
                input = 0;
                lcd_gotoxy(15,0);
                lcd_putsf("c");
                lcd_putsf(" ");

        }
        else if(input == 'd')
        {
                PORTB.3 = 1;
                delay_ms(100);
                input = 0;
                lcd_gotoxy(15,0);
                lcd_putsf("d");
                lcd_putsf(" ");
        }
        else if(input == 'e')
        {
                PORTB.4 = 1;
                delay_ms(100);
                input = 0;
                lcd_gotoxy(15,0);
                lcd_putsf("e");
                lcd_putsf(" ");
        }
        else if(input == 'f')
        {
                PORTB.5 = 1;
                delay_ms(100);
                input = 0;
                lcd_gotoxy(15,0);
                lcd_putsf("f");
                lcd_putsf(" ");
        }

      else if(input == ' ')
        {
            PORTB.0 = PORTB.1 = PORTB.2 = PORTB.3 = PORTB.4
= PORTB.5 = 0;
            input = 0;
        }
    }
```

B. Program pengiriman data antar mikrokontroler dari sisi *transmitter*.

Diagram alir untuk melakukan pengiriman data ke minimum sistem *receiver* terdapat pada gambar 3.16 dibawah ini.



Gambar 3.16. Diagram alir program pengiriman data ke minimum sistem *receiver*

Berikut potongan program pengiriman data dari sisi *transmitter* ke minimum sistem *receiver*

```
// Declare your global variables here
#define t_naik   PINB.2
#define t_turun  PINB.3
#define t_maju   PINB.4
#define t_mundur PINB.5
#define t_jepit  PINB.6
#define t_lepas  PINB.7

// ADC initialization
// ADC Clock frequency: 691.200 kHz
// ADC Voltage Reference: AREF pin
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE & 0xff;
```

```
ADCSRA=0x84;

// SPI initialization
// SPI disabled
SPCR=0x00;

// TWI initialization
// TWI disabled
TWCR=0x00;

while (1)
      {
      // Place your code here
          input = getchar();
          if (input == 'q')
          {
              putchar((unsigned int)read_adc(0));
              putchar((unsigned int)read_adc(1));
              //delay_ms(100);
              tombol();
              delay_ms(100);
          }
      }
}

void tombol()
{
    if(t_naik == 0)
        putchar('a');
    else if(t_turun == 0)
        putchar('b');
    else if(t_maju == 0)
        putchar('c');
    else if(t_mundur == 0)
        putchar('d');
    else if(t_jepit == 0)
        putchar('e');
    else if(t_lepas == 0)
        putchar('f');
    else
       putchar(' ');
}
```

C. Program penerapan kontroler PID

```
 {
      // Place your code here

        sp_ingin = read_adc(0);
        if (sp_ingin >200)
            sp_ingin=200;

        itoa(sp_ingin,tampung);
        lcd_gotoxy(0,0);
        lcd_puts(tampung);
        lcd_putsf("      ");
        while(f_lok == 0);
```

```
            putchar((unsigned char)rpm);
            f_lok=0;
            sp_aktual = rpm;
            itoa(sp_aktual,tampung_2);
            lcd_gotoxy(0,1);
            lcd_puts(tampung_2);
            lcd_putsf("           ");

            eror_baru = (sp_ingin - sp_aktual);

            hasil_p = kp*(eror_baru - eror_lama);
            hasil_i = ki*(eror_baru + eror_lama)/2;
            hasil_d = kd*(eror_baru - (2*(eror_lama +
eror_lama_2)));

            rmot_baru = rmot_lama + (int)hasil_p +
(int)hasil_i + (int)hasil_d;

            if (rmot_baru <= 0)
                motor_utama_kiri = 0;
            else if (rmot_baru > 255)
                motor_utama_kiri = 255;
            else
                motor_utama_kiri = (int)rmot_baru;

            ftoa(rmot_baru, 0,tampung_1);
            lcd_gotoxy(8,0);
            lcd_puts(tampung_1);
            lcd_putsf("     ");

            rmot_lama = rmot_baru;
            eror_lama_2 = eror_lama;
            eror_lama = eror_baru;
        }
}
```