BAB III

METODE PENELITIAN DAN PERANCANGAN SISTEM

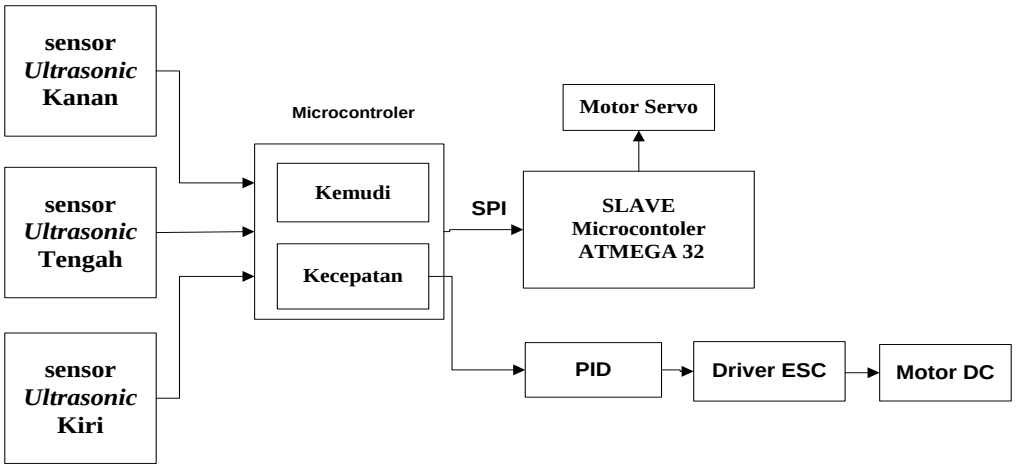
3.1 Metode Penelitian

Metode penelitian yang digunakan adalah studi kepustakaan dan penelitian laboratorium. Studi kepustakaan dilakukan sebagai penunjang yang berupa data-data literatur dari masing-masing komponen, informasi dari internet dan konsep-konsep teoretis dari buku-buku penunjang.

Penelitian laboratorium berupa perancangan perangkat keras, perancangan perangkat lunak, uji coba dan pengambilan data laboratorium. Perancangan perangkat keras dan perangkat lunak akan dibahas detil pada sub bab 3.3 dan 3.4. Sedangkan uji coba akan dilakukan terhadap minimum sistem ATTmega 32, rangkaian ***power***, dan. Untuk uji coba sensor *Ultrasonic Distance* akan diperoleh data berupa tegangan digital yang tidak linier terhadap jarak yang diukur. Data ini akan dikalibrasi yang kemudian akan digunakan dalam PID *controller*.

3.2 Rancangan Penelitian

Pembahasan proses rancang bangun *obstacle avoidance robot* dijelaskan pada diagram blok seperti Gambar 3.1.



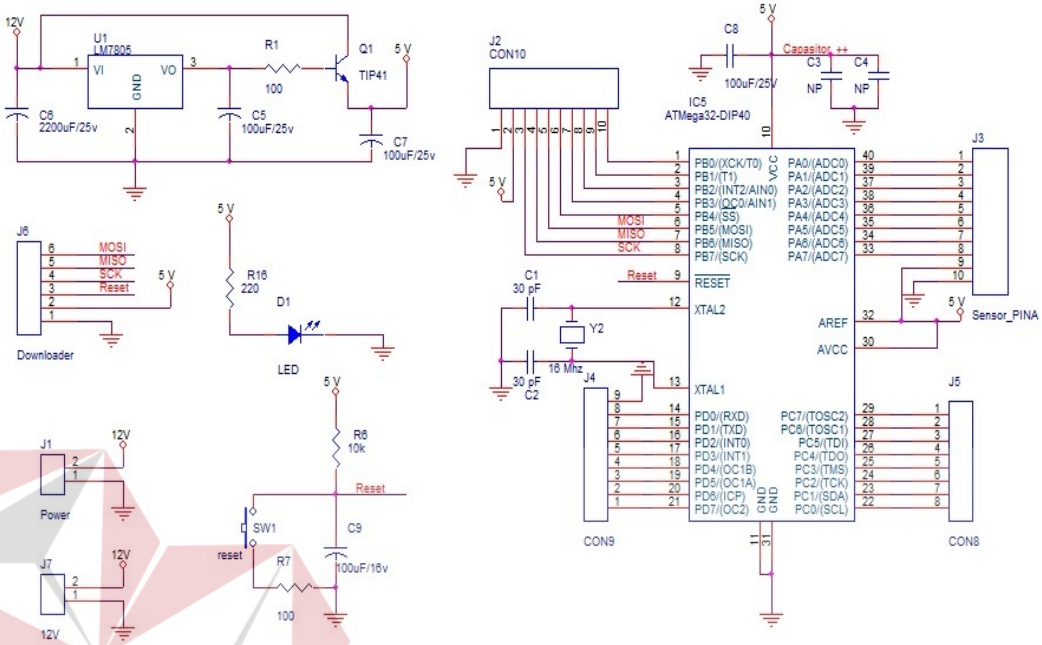
Gambar 3.1 Diagram blok *obstacle avoidance robot.*

Sensor *Ultrasonic Distance* memberi inputan pada Mikrokontroler Atmega, untuk mendeteksi adanya halangan benda di depannya pada jarak yang telah ditentukan. Set point digunakan untuk menentukan besarnya nilai tegangan yang berkorelasi dengan jarak benda terhadap robot. Apabila benda berada pada jarak yang terlalu dekat dengan set point yang telah ditentukan maka robot akan mengerem otomatis dan motor bergerak mundur. Begitu pula sensor *Ultrasonic Distance* kanan dan kiri digunakan untuk mendeteksi adanya halangan disebelah kanan dan kiri robot dan kemudian hasil pembacaanya akan diolah oleh mikrokontoler master Atmega 32.

Data *pulse* dari sensor *Ultrasonic Distance* dari Atmega master akan di transfer ke Atmega slave dengan menggunakan media *Serial peripheral Interface* (SPI), data tersebut kemudian diolah menjadi sinyal frequensi untuk memberikan perintah pada moto driver *Electronic Speed Control* (ESC) untuk mengatur kecepatan motor. Atmega slave dipergunakan untuk menggerakkan servo, pemutar kemudi.

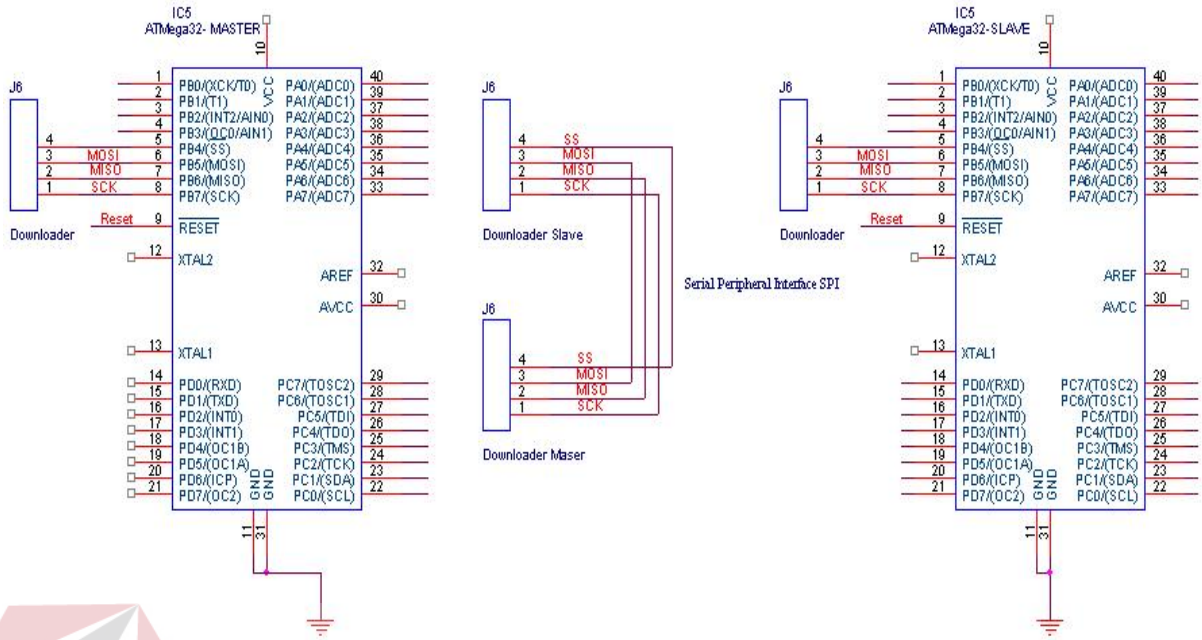
3.3 Perancangan Perangkat Keras

3.3.1 Rangkaian Minimum Sistem Atmega 32



Gambar 3.2 *minimum sistem Microcontroler* Atmega 32.

Minimum sistem microcontroler terdiri dari komponen-komponen dasar yang dibutuhkan oleh suatu *microcontroler* untuk dapat berfungsi dengan baik. Pada umumnya, suatu mikrokontoler membutuhkan dua elemen (selain *power supply*) untuk berfungsi: Kristal Oscillator (XTAL), dan Rangkaian *reset*. Analogi fungsi Kristal Oscillator memompa data. Fungsi rangkaian *reset* adalah untuk membuat mikrokontroler memulai kembali pembacaan program, hal tersebut dibutuhkan pada saat mikrokontroler mengalami gangguan dalam meng-eksekusi program. Pada sistem minimum AVR khususnya Atmega 32 terdapat elemen tambahan (*optional*), yaitu rangkaian pengendalian *gnd vcc dan vref (Tegangan Referensi)*. Pada konektor ISP untuk mengunduh (*download*) program ke *microcontroler*.



Gambar 3.3 *minimum sistem microcontroler* Atmega 32 Komunikasi SPI.

Beberapa tahapan dalam pengiriman data *Serial Peripheral Interface (SPI)* diantaranya pin *Slck* dari minimum master ke mimimum slave yang dipergunakan untuk clock. Pada pin *Mosi* jalur data akan dikirim dari master ke slave, begitu pula sebaliknya data pada slave juga dikirim ke master melalui pin *Miso.* Data pada minimum sistem master akan dikirim jika tegangan 0 v komunikasi membuka, bila mendapatkan tegangan 5 v komunikasi pada pengiriman data menuju ke minimum sistem slave akan gagal.

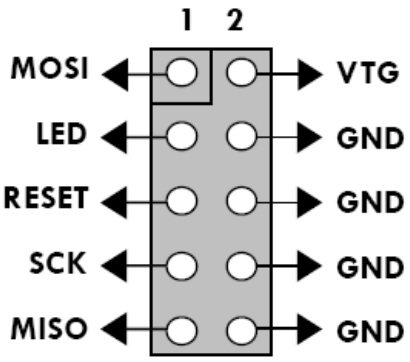
a. Program Downloader

Untuk melakukan proses *download* program, yaitu file dengan ekstensi ".hex" digunakan perangkat bantu AVR USB ISP seperti pada gambar 3.4 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 3.1 dan Gambar 3.4.

Gambar 3.4 AVR USB ISP.

Tabel 3. 1 Tabel Fungsi PIN

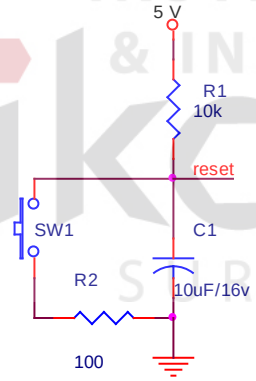
NAMA	NO.PIN	I/O	KETERANGAN
VTG	2	-	Catu daya dari project board (2.7 – 5.5 V)
GND	4, 6, 8,10	-	Titik referensi
LED	3	Output	Sinyal *control* untuk LED atau *multiplexer* (opsional)
MOSI	1	Output	Command dan data dari AVR USB ISP mkII ke target AVR
MISO	9	Input	Data dari target AVR ke AVR USB ISP mkII
SCK	7	Output	Serial *clock*, dikendalikan oleh AVR USB ISP mkII
RESET	5	Output	Reset, dikendalikan oleh AVR USB ISP mkII



Gambar 3. 5 PINOUT Connection .

b. Rangkaian Reset

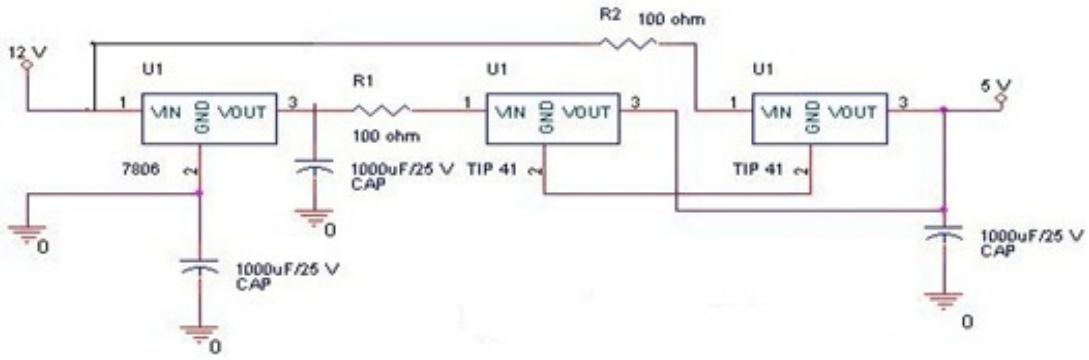
Pin reset pada *microcontroller* adalah pin (kaki) 1. Reset dapat dilakukan secara manual atau otomatis saat power dihidupkan (Power reset ON).



Gambar 3.6 Rangkaian Reset.

Reset terjadi dengan adanya logika 1 selama minimal 2 *machine cycle* yang diterima pin reset dan akan bernilai *low*. Pada saat reset bernilai *low*, *microcontroller* akan melakukan reset program yang ada di dalam *microcontroller* dan mengakhiri semua aktivitas pada *microcontroller*.

3.3.2 Rangkaian Power



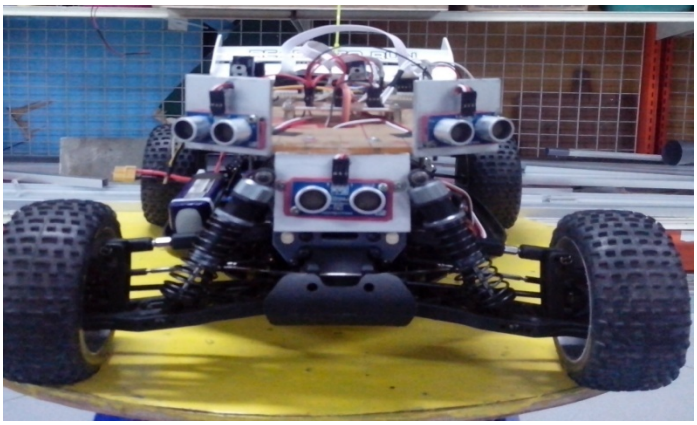
Gambar 3.7 Rangkaian Power.

Sumber tegangan input dari baterai 12 volt akan masuk ke transistor, tegangan langsung diturunkan dengan transistor 7806 sehingga tegangan menjadi 5,5 volt dengan arus 1 A. Output dari transistor 7806 akan masuk ke resistor 100 Ω untuk mengurangi 0,3 A, selanjutanya arus masuk ke input kaki *base* Tip 41. Pada kaki *collector* Tip 41 yang dipasang secara pararel, sehingga outputnya arus menjadi 8 A, karena pada tiap-tiap Tip 41 mempunyai arus 4 A pada outputanya. Kapasitor di rangkaian power untuk menyimpan daya saat baterai dari sumber tegangan mati.

3.4 Perancangan Arsitektur Sistem

3.4.1 Pengendali Sistem Kemudi

Perancangan arsitekturnya dapat dilihat seperti gambar 3.8. Pada gambar 3.8 adalah gambar robot tampak depan. Untuk sensor Ping *Ultrasonic Distance* diletakkan di bagian depan. *Ultrasonic Distance* yang sebelah kiri dan kanan dipasang menyerong 30° dan letaknya lebih tinggi dibandingkan dengan sensor bagian depan.



Gambar 3.8 Robot Tampak Depan.

3.5 Perancangan Sistem Kerja Motor

Perancangan sistem kerja motor *Brushless* terdapat beberapa tahadapan kondisi motor. Kondisi motor tersebut terbagi 3 bagian, bagian 1 yaitu proses motor diam, dibagian ke 2 motor maju, dan bagian ke 3 motor mundur.

3.5.1 Proses Motor Diam

Untuk dapat mencari motor diam dapat dilakukan dengan cara mencari data yang dikirimkan dari *Electronic Speed Control* (ESC) menuju motor *Brushless* dan data tersebut berupa *frekuensi*. Dalam menentukan frekuensi berapa dia berjalan maka di ukur menggunakan *Osiloskop*. Pada proses motor diam berada pada *frekuensi* Th 1500 µs dan Tl 17000 µs. Data hasil uji coba pengukuran lebar *frekuensi* ditunjukkan dalam tabel 3.2.

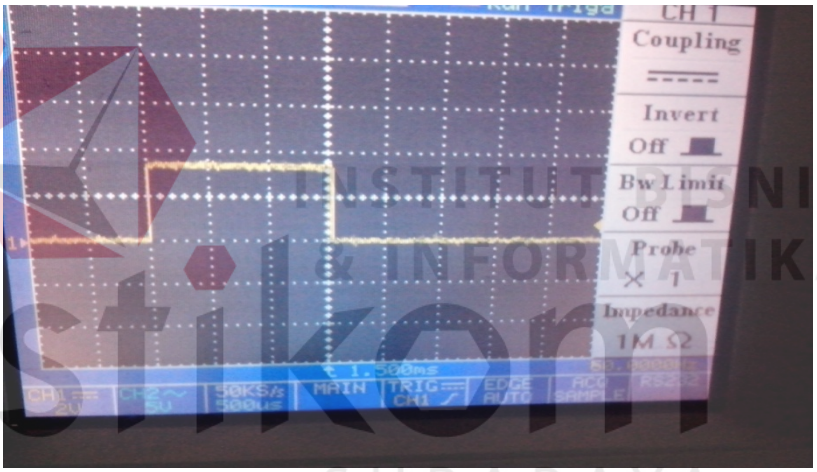
3.5.2 Proses Motor Maju

Pada proses motor maju, data *frekuensi* yang terdapat pada *Electronic Speed Control* (ESC) akan dikirim ke motor *Brushless*. Data dari *Electronic Speed Control* (ESC) menuju motor akan dilakukan proses tab data pada *remote control* mobile Rc, tuas *remote* ditekan ke belakang

sehingga data sinyal dari *reciver* akan masuk pada *Osiloskop*. Data hasil uji coba pengukuran lebar *frekuensi* ditunjukkan dalam tabel 3.2.

3.5.3 Proses Motor Mundur

Sama halnya pengujian proses motor maju, pada motor mundur pengujian *Electronic Speed Control* (ESC) menuju motor akan dilakukan proses tab data pada *remote control* mobile Rc, tuas *remote* ditekan ke depan sehingga data sinyal dari *reciver* akan masuk pada *Osiloskop*. Setelah dilakukan proses tab data dapat dilihat lebar *pulsa* motor dapat dilhat pada gambar 3.9.



Gambar 3.9 Pengujian Hasil lebar *frekuensi* pada *osiloskop*.

Pada pengujian sistem kerja motor diperoleh hasil nilai data yang ada pada *osiloskop* berupa *frekuensi* motor diam, motor maju, dan motor mundur. Dapat dilihat pada tabel 3.2.

Tabel 3.2 Perhitungan *frekuensi* data pada *Osiloskop*

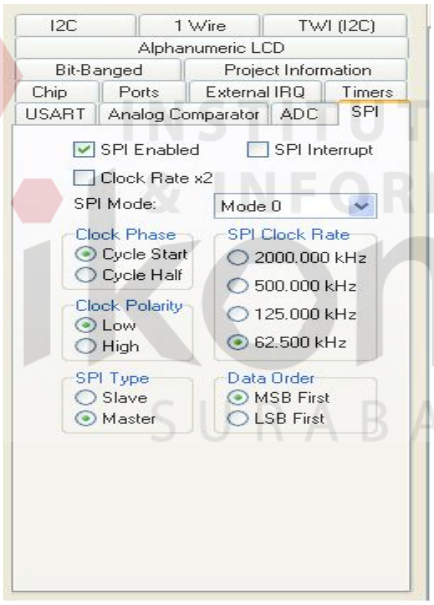
Th (µs) (Periode)	TL (µs) (periode)	Nilai frekuensi (ms)	Keterangan
1500	**17000**	**0,00072549**	Diam
1550	**16950**	**0,000704158**	Maju
1551	**16949**	**0,000703746**	Maju
1552	**16948**	**0,000703334**	Maju
1553	**16947**	**0,000702922**	Maju
1554	**16946**	**0,000702512**	Maju
1555	**16945**	**0,000702101**	Maju
1556	**16944**	**0,000701691**	Maju
1557	**16943**	**0,000701282**	Maju
1558	**16942**	**0,000700873**	Maju
1559	**16941**	**0,000700465**	Maju
1560	**16940**	**0,000700058**	Maju
1250	**17250**	**0,000857971**	Mundur
1240	**17260**	**0,000864389**	Mundur
1230	**17270**	**0,000870912**	Mundur
1220	**17280**	**0,000877543**	Mundur
1210	**17290**	**0,000884283**	Mundur
1200	**17300**	**0,000891137**	Mundur

3.6 Perancangan Perangkat Lunak

Selain hardware yang diperlukan pada perancangan dan pembuatan pada penelitian ini juga diperlukan software / program pada *microcontoler*, serta komunikasi *microcontoler* master dan Slave agar dapat bekerja sesuai dengan fungsinya.

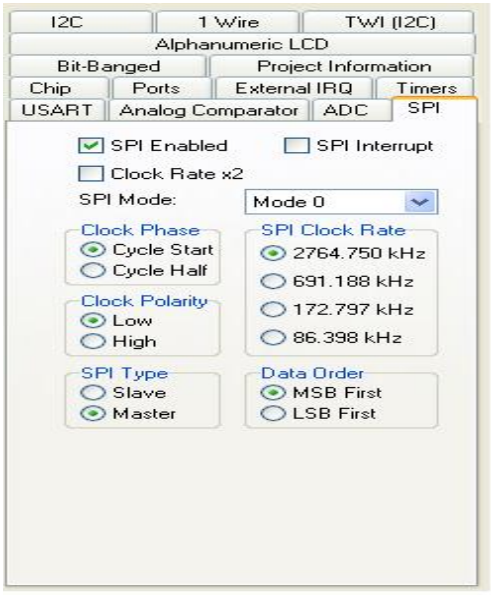
3.6.1 Perancangan Program Pada Mikrokontoler Master Slave

Perancangan Program di aplikasi *cvavr*, buka program baru gunakan *CodeWizardavr* pada option pilih Spi enabled serta pilih Spi master.



Gambar 3. 10 Setting Konfigurasi Master Cvavr master.

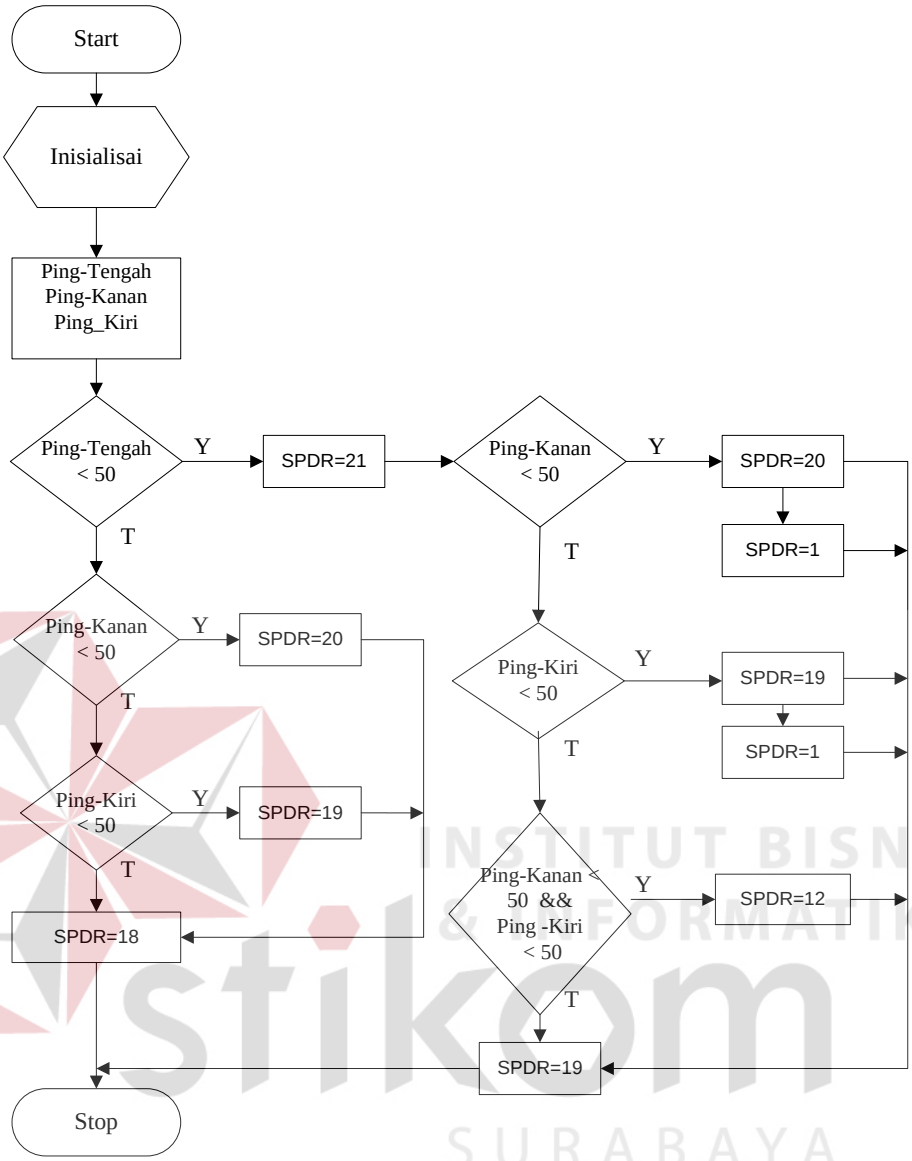
Pada gambar 3.11 buka program baru gunakan *CodeWizardavr* pada option pilih *Spi enabled* serta pilih *Spi slave*. Namun pada *Spi slave* di tambah *clock rate* untuk kecepatan pengeriman data dari *microcrontoler master* ke *slave.*



Gambar 3.11 Setting Konfigurasi Master Cvavr Slave.

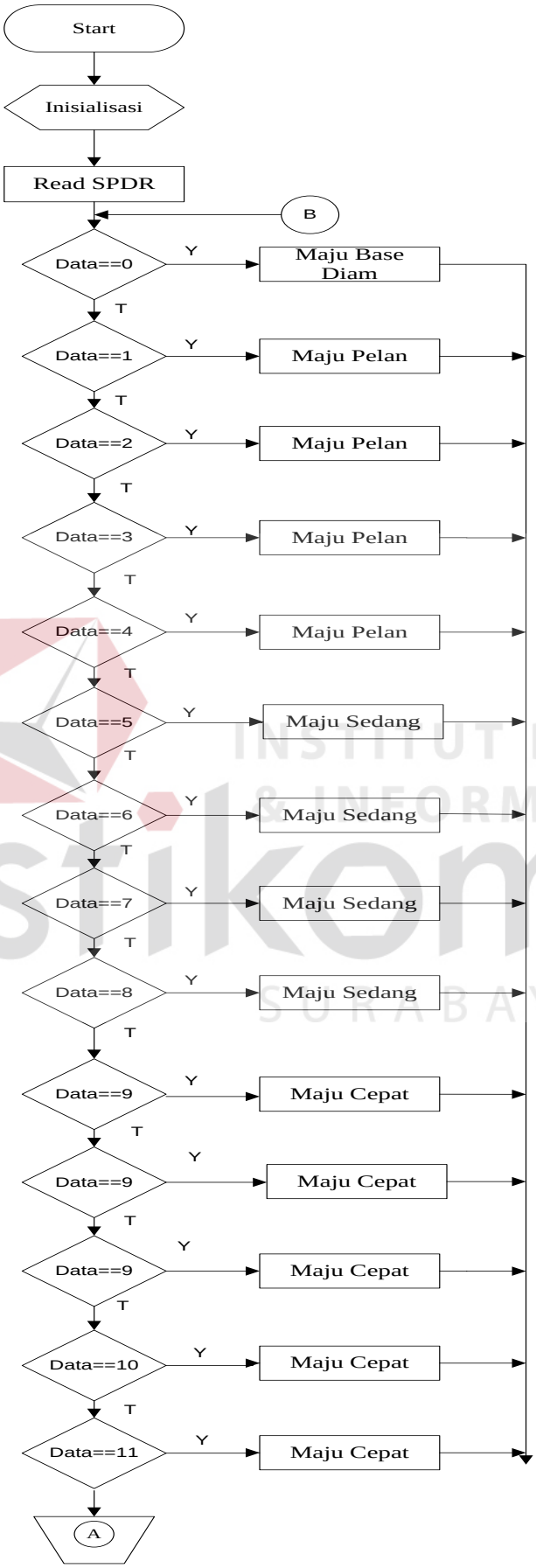
3.6.2 Perancangan Program Pada mikrokontroler

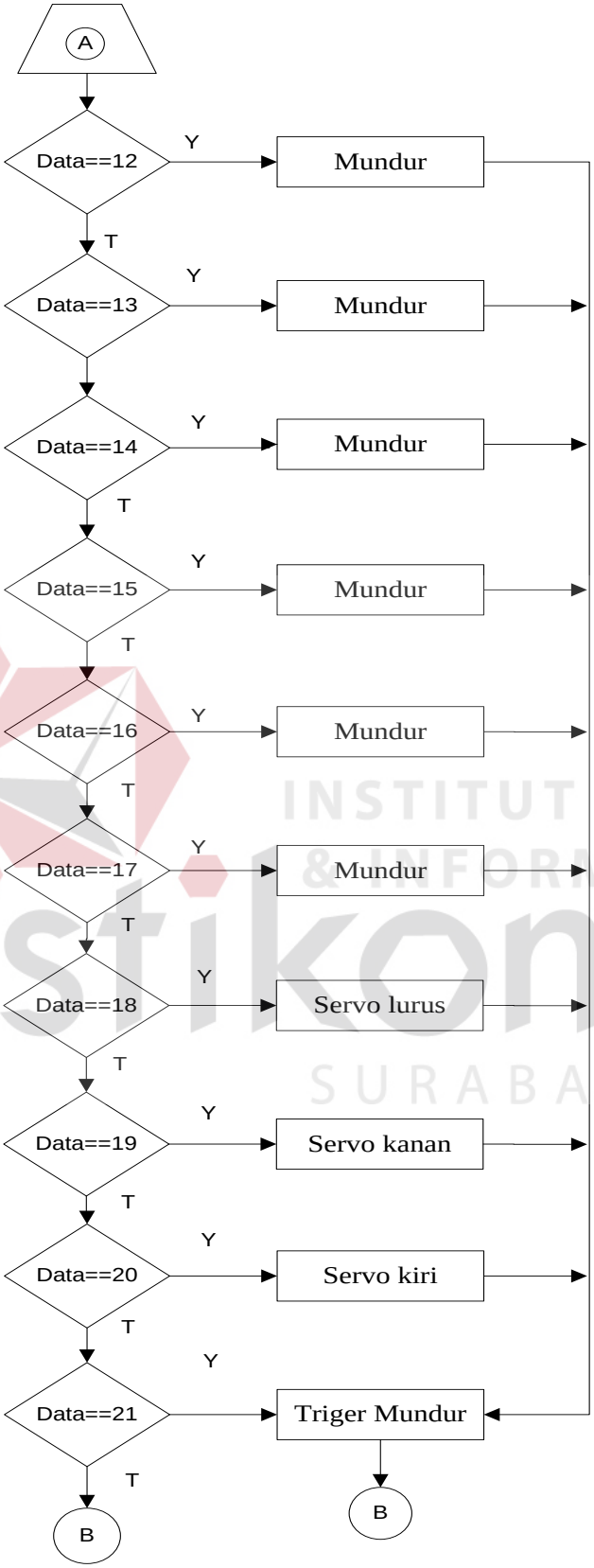
Penjelasan dari diagram flow pada gambar 3.12 adalah sebagai berikut: Sensor *Ultrasound Distance* kanan, kiri dan tengah secara terus menerus mendeteksi jarak. Sensor tengah mendapat prioritas utama dalam mendeteksi halangan di depan robot, apabila sensor tengah mendapatkan halangan pada jarak yang terlalu dekat (< 50) maka pada *microcontoler master* akan mengirimkan data *Serial Peripheral Interface* (SPI) ke *microcontoler slave* untuk merespon dengan memberikan perintah mundur. Apabila kondisi pada sensor kanan < 50 maka data akan diolah *microcontoler slave* agar dapat merespon kemudi bergerak ke kiri, dan sebaliknya jika halangan < 50 terdapat di sisi sebelah kiri robot maka data akan langsung dikirim olah *microcontoler slave ke microcontoler master* untuk kemudian memberikan respon bergerak ke kanan.



Gambar 3.12 *Flowchart* program kemudi

Pada gambar 3.12 proses *Flowchart* sistem kemudi robot *Obstacle Avoidance* dirancang berdasarkan jarak robot terhadap halangan. Robot akan bergerak bebas ke arah kanan maupun kiri untuk dapat menghidari halangan. Saat robot mulai dijalankan sensor tengah akan langsung mendeteksi halangan tersebut berada, sensor tengah akan merespon sistem kemudi. Pada sistem ini robot akan terus berjalan serta mendeteksi halang di depan samping kanan dan samping kiri tanpa adanya jalur lintasan robot dari titik satu ke titik yang lainya.





Gamabar 3.13 *Flowchart* program laju kecepatan

Dari proses diagram *flow* pada gambar 3.13 sebagai berikut: data *microcontoler minimum sistem master* akan dikirim melalui kabel *serial peripheral interface (SPI)* ke *microcontoler minimum sistem master.* Pada proses diagram flow pada gambar 3.13 *microcontroler minimum sistem master* data pada *read SPI Data Register* (*SPDR*) akan dipergunakan untuk membaca/ mengenali proses pengiraman data maupun penerima. Dalam penginputan data pada *microcontoler* dibagi beberapa kondisi motor diam, motor pelan, sedang, maju cepat, mundur dan juga pengaturan servo. Pada *microcontoler* data 0 adalah motor base diam, data 1 motor maju pelan, data 5 motor melaju sedang, data 9 motor akan melaju cepat, sedangkan data 12 dipergunakan untuk mundur jika halangan tersebut terlalu dekat dengan robot *base*. Sedangkan pada proses kemudi data 18, 19 servo ke kanan dan ke kiri. Setelah proses tersebut terpenuhi sampai data pada *SPI Data Register* (*SPDR*) maka langsung akan diolah ke *microcontoler minimum sistem master.* Pada sistem *Flowchart* robot akan berhenti berjalan bila tombol *off* pada rangkaian robot dimatikan.