IN SYSTEM PROGRAMMING AVR MENGGUNAKAN KONEKSI *BLUETOOTH*

Rezky Ardiansyah[1)] Andrew Joewono[2)] Diana Lestariningsih[2)]
E-mail: rezky.ardiansyah@rocketmail.com

ABSTRAK

Salah satu produk ISP yang berada di pasaran antara lain: DT-HIQ AVR USB ISP, dan USB ASP masih menggunakan kabel sebagai transmisi datanya. Sehingga jika ingin men-download suatu program ke chip mikrokontroler harus berdekatan antara kompuer dan mikrokontroler target. Kondisi ini akan menjadi masalah jika mikrokontroler target bergerak, seperti halnya mikrokontroler target yang berada pada badan robot. Jika program pada robot itu masih bermasalah, maka robot tersebut harus didekatkan kembali ke komputer untuk diprogram ulang. Oleh karena itu dibutuhkan alat untuk men-download program tanpa melalui media kabel. Dengan demikian akan memudahkan programmer dalam memprogram suatu device dan dapat menghemat tenaga, karena jika ingin memprogram ulang suatu device tidak perlu lagi mendekatkan device ke komputer programmer.

Pada protokol STK500, data yang dikirimkan berupa paket. Sehingga alat ini mempunyai fungsi untuk memeriksa paket data yang diterima. Setelah paket data benar, alat ini akan mengklasifikasikan perintah yang diterima. Alat ini berfungsi untuk men-download program ke mikrokontroler target. Jalur yang digunakan ialah pin MOSI, MISO, dan SCK.

Alat ini dapat menghemat tenaga programmer karena koneksi yang digunakan ialah menggunakan koneksi bluetooth (tanpa kabel). Area coverage alat ini mencapai sekitar 16 meter dari jarak komputer pengguna.

Kata Kunci: *in system programming AVR, bluetooth*, mikrokontroler

PENDAHULUAN

Salah satu produk *downloader* yang berada di pasaran antara lain *DT-HIQ AVR USB ISP*, dan *USB ASP* yang masih menggunakan kabel untuk mentransmisi data. Jika pengguna ingin men-*download* suatu program ke *chip* mikrokontroler harus berdekatan antara komputer dan mikrokontroler target karena keterbatasan panjang kabel. Kondisi ini akan menjadi masalah jika mikrokontroler target bergerak, salah satu contoh yaitu mikrokontroler target yang berada pada badan robot. Jika program pada robot masih terdapat masalah, maka robot tersebut harus didekatkan kembali ke komputer untuk diprogram ulang. Untuk mengatasi masalah tersebut dibutuhkan alat untuk men-*download* program tanpa melalui media kabel, sehingga akan memudahkan *programmer* dalam memprogram suatu *chip* mikrokontroler. Hal ini dapat menghemat waktu dan tenaga *programmer* jika membutuhkan untuk memprogram ulang suatu chip mikrokontroler. *Programmer* tidak perlu mendekatkan *device* ke komputer *programmer*.

Adapun permasalahan yang dihadapi penulis dalam pembuatan alat adalah sebagai berikut:

1. Bagaimana modul *bluetooth* EB500 dapat tersinkronisasi terhadap *bluetooth dongle* pada komputer pengguna;
2. Bagaimana realisasi protokol STK500 pada komunikasi antara pengguna dan *downloader* ke dalam program mikrokontroler;
3. Waktu yang dibutuhkan *downloader* untuk men-*download* suatu program ke mikrokontroler target dengan berbagai *variabel* sebagai pembandingnya;
4. Bagaimana tingkat kesesuaian program yang di-*download.*

Metode yang digunakan dalam perealisasian alat ini meliputi: studi literatur, perencanaan alat, pembuatan alat, pengujian alat, menarik simpulan hasil perancangan dan pengujian sistem.

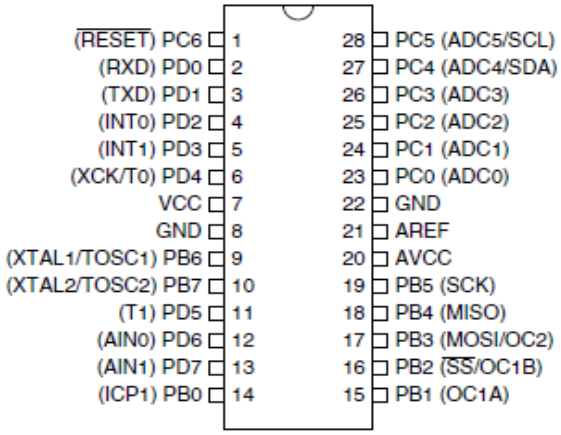
TINJAUAN PUSTAKA

Mikrokontroler AVR AT Mega8[2]

Mikrokontroler yang digunakan dalam penelitian ini adalah mikrokontroler *Alf and Vegard's Risc Processor* (*AVR*) *AT Mega8*, produk ini merupakan keluaran dari keluarga *ATMEL*. Mikrokontroler *AVR* sama halnya seperti keluaran *ATMEL* lainnya seperti 8051, yaitu memiliki 8 pin dalam 1 *port* yang dapat digunakan sebagai *input* atau *output* digital dan penggunaan pin (*MISO, MOSI, SCK, RESET*) untuk memasukkan *file* dalam bentuk heksadesimal dari komputer ke mikrokontroler. Mikrokontroler *AVR* memiliki arsitektur *RISC*

[1)] Mahasiswa di Fakultas Teknik Jurusan Teknik Elektro Universitas Katolik Widya Mandala Surabaya
[2)] Staf Pengajar di Fakultas Teknik Jurusan Teknik Elektro Universitas Katolik Widya Mandala Surabaya

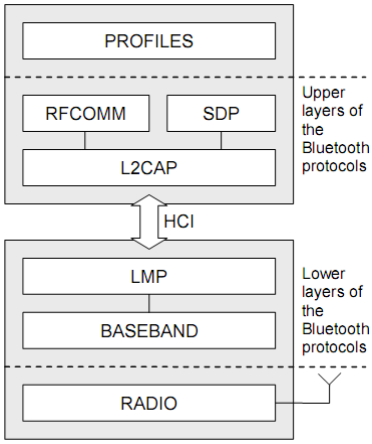
Reduce Instruction Set Computing (*RISC*) 8-bit, semua instuksi dikemas dalam kode 16-bit (16-*bits word*) dan sebagian besar dilakukan dalam 1 siklus *clock*. **Gambar 1** menunjukkan konfigurasi pin *AT Mega8*.



Gambar 1. Konfigurasi Pin *AT Mega8*

Komunikasi Data *Bluetooth*

Bluetooth didefinisikan sebagai sebuah arsitektur protokol ber-*layer* yang mengandung protokol inti, pengganti kabel dan protokol telepon serta protokol adaptasi. Pada bagian selanjutnya akan dibahas lebih lanjut mengenai masing–masing blok dari arsitektur protokol sebagaimana yang terlihat pada **Gambar 2.**



Gambar 2. Arsitektur Protokol *Bluetooth*

Layer paling dasar dari arsitektur *bluetooh* adalah *Bluetooth Radio*. *Bluetooth* beroperasi pada *Industrial Scientific and Medical (ISM) band* antara 2,40 sampai dengan 2,48 GHz. Terdapat tiga kelas dari *transmitter* yang dibedakan dari *output power* dan jangkauan transmisi yang dimiliki. *Bluetooth* radio menggunakan modulasi *Gaussian Frequency Shift Keying* (*GFSK*) dalam melakukan transmisinya. *Baseband layer* adalah lapisan fisik dan berada di atas *Bluetooth radio*. *Baseband layer* menangani masalah *channel*.

Untuk menghindari terjadinya interferensi dari sinyal lain yang berada pada *band* 2,4 GHz seperti *WLAN*, *Bluetooth* menggunakan prinsip frekuensi *hopping*. Setiap perangkat *bluetooth* mempunyai alamat *hardware* yang khusus dan sebuah *bluetooth clock*. Sebuah algoritma tertentu digunakan untuk menghitung frekuensi *hopping* yang digunakan berdasarkan pada *hardware address* dari perangkat yang menjadi *master* dan *clock*. Untuk komunikasi *duplex*, skema *Time Division Duplex* (*TDD*) digunakan sebagai sarana pengiriman dan penerimaan data. Di mana pada skema ini perangkat *master* melakukan transmisi pada *slot* waktu genap dan perangkat *slave* melakukan transmisi pada *slot* waktu ganjil. *Link Manager* berfungsi untuk menjalankan *link setup*, *authentication*, *link configuration* dan aspek lainnya dari hubungan radio antara *master* dan *slave*. Selain itu juga berfungsi untuk menemukan *link manager* lainnya dan melakukan komunikasi dengan menggunakan *link manager protocol*. Untuk memastikan *hardware* yang berbeda dapat saling mendukung, maka perangkat *bluetooth* menggunakan *Host Controller Interface* (*HCI*) sebagai tatap muka antara *host bluetooth* (contohnya komputer) dan *Baseband* dan *Link Manager*.

Logical Link Control and Adaption Protocol (*L2CAP*) adalah lapisan yang berada di atas *baseband protocol* dan melayani protokol yang di atasnya dengan kemampuan *protocol multiplexing* dan *packet segmentation* serta *reassembly* (*SAR*). Fungsi *SAR* diperlukan untuk mendukung protokol yang menggunakan paket data yang lebih besar daripada yang dapat didukung oleh *baseband*. Paket *L2CAP* yang besar dibagi ke dalam beberapa paket *baseband* yang lebih kecil sebelum dipancarkan dan paket-paket tersebut akan disusun kembali setelah diterima menjadi paket *L2CAP*. *Service Discovery Protocol* pada dasarnya berfungsi untuk menyediakan perangkat (*tools*) yang dibutuhkan oleh sebuah aplikasi untuk memperoleh informasi dari sebuah perangkat *bluetooth*, layanan *bluetooth* apa saja yang tersedia dan untuk menentukan karakteristik dari layanan yang tersedia. *Port* serial merupakan salah satu dari layanan komunikasi yang sering digunakan.

RFCOMM adalah pengganti protokol kabel yang termasuk di dalam spesifikasi *bluetooth*. *RFCOMM* melakukan emulasi *port serial* RS-232 ke dalam protokol *L2CAP* dan menggantikan fungsi kabel *serial*. *RFCOMM*

dapat menyediakan sampai dengan 60 *port* di antara dua perangkat *bluetooth*. *Bluetooth profile* digunakan untuk memberikan gambaran untuk implementasi dari sebuah fungsi yang dapat dilakukan. *Bluetooth profile* ditujukan untuk mengurangi masalah yang mungkin timbul dari perbedaan produk yang diproduksi oleh perusahaan yang berbeda. *Profile* yang dimiliki *bluetooth* sampai saat ini antara lain: *basic imaging, basic printing, cordless telephony, dial-up networking, file transfer, handsfree, headset, serial port,* dan *scynchronization.*

Embedded Blue 500[3]

Embedded Blue 500 (*EB500*) adalah modul *bluetooth* yang digunakan untuk menghubungkan mikrokontroler dengan *PC*. *Embedded Blue 500* mempunyai 2 modus operasi yaitu *command mode* dan *data mode*. Pada *command mode*, *EB500* akan menerima perintah serial yang diberikan, yang berkaitan dengan proses pengaturan pada *EB500* (perintah serial tersebut tidak dikirimkan), sedangkan pada *data mode*, *EB500* akan berfungsi sebagai jalur komunikasi data serial antara *device bluetooth* yang berbeda (meneruskan pengiriman maupun penerimaan data serial). Gambar *EB500* dapat dilihat pada Gambar 3.

Gambar 3. Modul *EB 500*

Protokol STK 500[4]

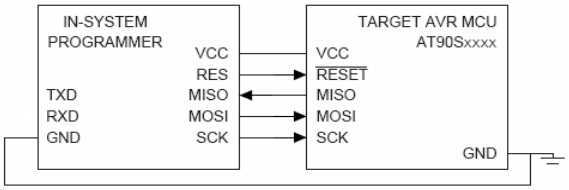
Protokol STK 500 digunakan untuk mengirimkan data dari *PC* ke mikrokontroler yang berkaitan dengan pemrograman suatu mikrokontroler. *Software compiler* seperti *CodeVisionAvr* telah menggunakan protokol STK 500. Protokol ini menggunakan *baud rate* 115200, 8 bit data, 1 bit stop, dan tidak menggunakan *parity* bit. STK 500 memiliki format pesan sebagai berikut: *MESSAGEE_START, SEQUENCE_NUUMBER, MESSAGE_SIZE, TOKEN, MESSAGE_BODY, CHECK_SUM*. Penjelasan dari masing-masing format dari protokol STK 500 disajikan pada Tabel 1.

Tabel 1. Penjelasan Fungsi Format Dari Protokol STK 500

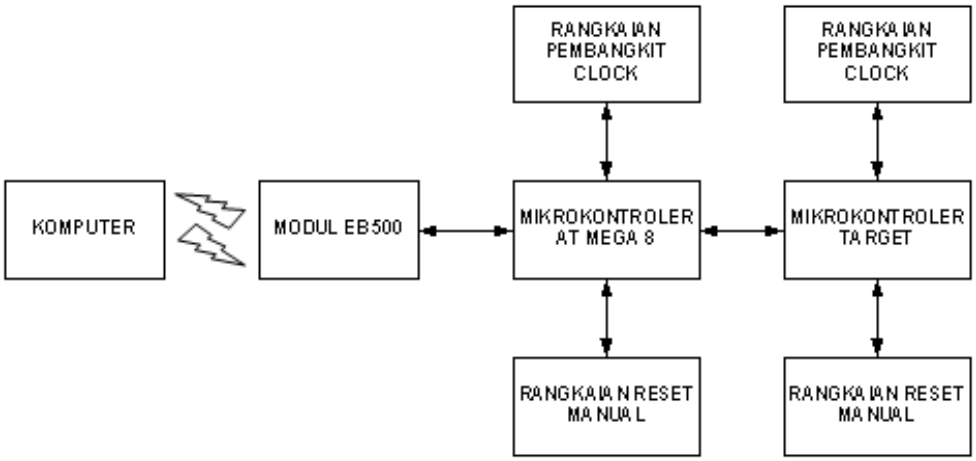
Name Parameter	Ukuran	Deskripsi
MESSAGE_ START	1 bit	Selalu bernilai 0x1B
SEQUENCE _NUMBER	1 bit	Nilainya akan bertambah satu setiap pengiriman pesan. Kembali ke 0 setelah 0xFF
MESSAGE_ SIZE	2 bit, MSB yang pertama	Ukuran dari *message_body*
TOKEN	1 bit	Selalu bernilai 0x0E
MESSAGE_ BODY	Message_ size bit	Bernilai dari 0 sampai 65535 byte
CHECKSUM	1 bit	Semua pesan di XOR kan

Protokol In System Programming[5]

Protokol In System Programming ialah suatu protokol yang digunakan untuk memprogram atau memprogram ulang mikrokontroler. Jalur yang digunakan ialah jalur *SPI* yang terdiri dari *Serial clock* (*SCK*), *Master In-Slave Out* (*MISO*), dan *Master Out-Slave In* (*MOSI*). Ketika melakukan pemrograman suatu mikrokontroler, *programmer* harus dioperasikan sebagai *master* dan mikrokontroler target dioperasikan menjadi *slave*. *Programmer* menyediakan *clock* untuk keperluan komunikasi pada pin *SCK*. Setiap pulsa pada pin *SCK* mentransferkan 1 bit data dari *programmer (Master)* menuju target *(slave)* pada pin *MOSI*. Secara bersamaan, setiap pulsa pada pin *SCK* mentransfer 1 bit data dari target *(Slave)* menuju *programmer (master)* pada pin *MISO*. Gambar koneksi antara *downloader* dan mikrokontroler target dapat dilihat pada Gambar 4.



Gambar 4. Koneksi *Programmer* dan Mikrokontroler Target



Gambar 5. Diagram blok sistem secara keseluruhan

Agar komunikasi antara *ISP* dan mikrokontroler target dapat berjalan dengan baik, *Ground* (*GND*) antara *ISP* dan mikrokontroler target harus bergabung menjadi satu. Untuk men-*download* suatu program, *ISP* akan men-*setting* pin *RST* mikrokontroler target menjadi *Low* (0 Volt). Setelah pin *Reset* bernilai 0, mikrokontroler target memasuki *mode programming* dan *Serial Peripheral Interface* (*SPI*) diaktifkan sehingga mikrokontroler target tersebut telah siap untuk menerima instruksi dari *programmer*.

Format perintah terdiri dari 4 bit, bit pertama terdiri dari kode perintah, pemilihan operasi dan target memori. Bit kedua dan ketiga terdiri dari alamat pada memori yang telah dipilih. Bit keempat terdiri dari data yang akan dikirimkan ke mikrokontroler target. Ada beberapa perintah pada protokol *ISP* ini antara lain, *Enable Memory Access, Device Code, Flash Program Memory Access, EEPROM Data Memory Access, Lock Bits Access,* dan *Chip Erase Operation.*

METODE PENELITIAN

Perancangan dan Pembuatan Alat

Adapun diagram blok dari alat ini dapat dilihat pada Gambar 5. Komputer digunakan sebagai media perantara antara pengguna dengan sistem dengan menggunakan bantuan *software compiler* di mana perintah dikirimkan /diterima dengan menggunakan sarana komunikasi *bluetooth* melalui *bluetooth adapter/dongle* ataupun internal *bluetooth* yang terdapat pada *PC*. *EB500* adalah modul *bluetooth* yang digunakan untuk menerima data dari *PC* untuk diteruskan ke mikrokontroler maupun sebaliknya, mengirimkan dari mikrokontroler ke *PC*.

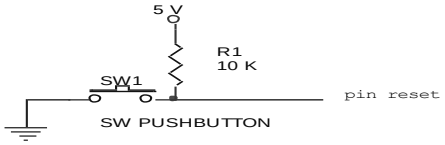
Mikrokontroler digunakan sebagai bagian utama sistem yang mengontrol kerja *In System Programming* untuk memprogram mikrokontroler target sesuai dengan perintah dari *PC*, dalam hal ini rangkaian mikrokontroler yang dimaksud ialah rangkaian minimum sistem *ATMega8*. Mikrokontroler target ialah mikrokontroler yang hendak di program dengan menggunakan protokol *In System Programming.*

Alat ini tidak membutuhkan *supply* 5 V, karena alat ini telah dirancang agar setelah terkoneksi dengan mikrokontroler target dapat mengambil *supply* dari rangkaian mikrokontroler target yang akan di *download* programnya.

Perancangan elektronika pada alat ini sebagai berikut:

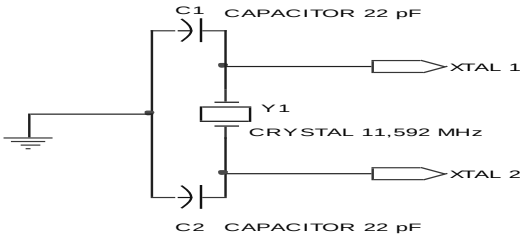
1. Rangkaian mikrokontroler *ATMega8*;
2. Rangkaian pembangkit *clock*;
3. Rangkaian *reset* manual.

Rangkaian minimum sistem mikrokontroler *ATMega8* terdiri dari rangkaian pembangkit *clock* dan *reset* manual. Pada rangkaian *reset* manual terdapat beberapa komponen yaitu resistor 10 K yang berfungsi untuk *pull up* pin reset mikrokontroler dan tombol reset. Skematik rangkaian reset manual dapat dilihat pada Gambar 6.



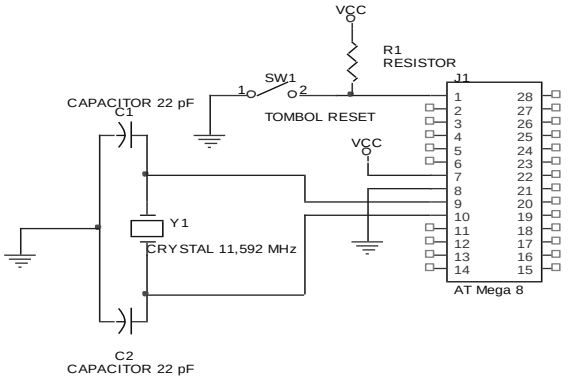
Gambar 6. Rangkaian Reset Manual

Sedangkan pada rangkaian pembangkit *clock* terdapat beberapa komponen, antara lain : kristal 11,592 MHz dan 2 buah kapasitor 22 pF. Bekerjanya sebuah mikrokontroler tergantung pada besarnya frekuensi kristal yang digunakan, semakin tinggi frekuensi kristal yang digunakan, maka semakin cepat kerja dari mikrokontroler. Sedangkan pemilihan frekuensi *clock* sebesar 11,592 MHz dikarenakan mikrokontroler menggunakan *baudrate* sebesar 115200 bps. Rangkaian pembangkit *clock* dapat dilihat pada Gambar 7 di bawah ini.



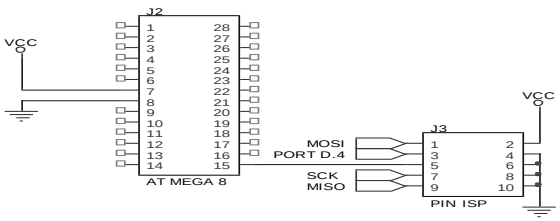
Gambar 7. Rangkaian Pembangkit *Clock*

Sehingga keseluruhan rangkaian minimum sistem *ATMega8* dapat dilihat pada Gambar 8 dibawah ini.



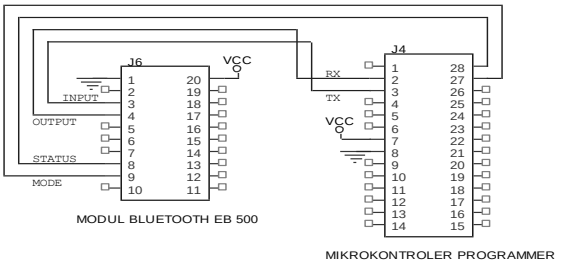
Gambar 8. Rangkaian Minimum Sistem *ATMega8*

Fungsi dari alat ini untuk memprogram mikrokontroler target, sehingga *header programming* tersebut juga dikoneksikan ke mikrokontroler target tetapi yang berbeda *reset* mikrokontroler target harus dikontrol oleh suatu pin mikrokontroler *programmer*. Karena untuk memasuki mode programming, mikrokontroler target harus diatur bernilai 0 (*low*). Untuk melakukan hal tersebut dapat dikontrol oleh mikrokontroler *programmer*. Konfigurasi pin dapat dilihat pada Gambar 9 di bawah ini.



Gambar 9. Konfigurasi Pin *Header Programming* Untuk Mikrokontroler Target

Komunikasi *bluetooth* yang digunakan ialah menggunakan *profil bluetooth* sebagai *serial port*. Modul *EB 500* yang digunakan untuk alat ini telah menggunakan *profil serial port*. Sehingga dibutuhkan komunikasi serial antara modul *bluetooth* dengan mikrokontroler. Koneksi antar kedua bagian dapat dilihat pada Gambar 10.



Gambar 10. Koneksi Antara Bluetooth Dengan Mikrokontroler

Pada perancangan *software* digunakan *software compiler WinAVR* sebagai *compiler*-nya. Pada perancangan awal ialah mengatur register-register yang diperlukan untuk komuniasi serial antara modul *EB500* dan mikrokontroler *ATMega8*. *ATMega8* memiliki fasilitas komunikasi serial yang terleak pada pin *RX* dan *TX*. Pin *RX* berfungsi untuk menerima data dari luar, sedangkan pin *TX* berfungsi untuk mengirimkan data dari mikrokontroler menuju keluar. Untuk menggunakan fasilitas komunikasi serial ini ada beberapa pengaturan register-register yang terkait. Pada penelitian ini komunikasi serial yang digunakan menggunakan *baudrate* 115200 bps dengan parameter 8 bit data, 1 stop bit, dan tidak menggunakan bit *parity*. Pemilihan *baudrate* ini disesuaikan dengan protokol STK 500 yang memerlukan komunikasi data dengan *baudrate* 115200 bps, sehingga komunikasi serial mikrokontroler menggunakan 115200 bps agar terjadi komunikasi antara 2 komponen tersebut. Register-register yang harus diatur ialah sebagai berikut:

```
// Set baud rate
UBRRH=0x00;
UBRRL=0x05;
// Set frame format to 8 data bits, no parity, 1 stop bit
UCSRC = 0x86;
// Enable receiver and transmitter
UCSRB = 0xD8;
```

Pengaturan register-register tersebut harus diesuaikan dengan protokol pengiriman yaitu protokol STK500. Pada protokol STK500 komuniksasi serial yang digunakan harus

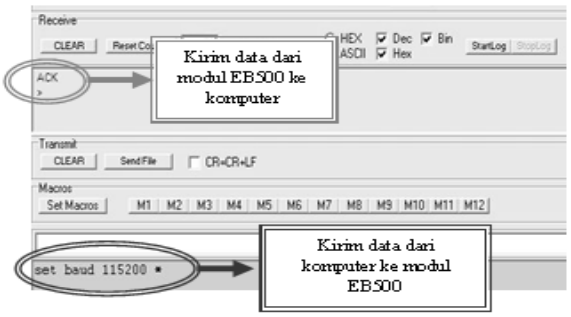
menggunakan *baudrate* 115200 bps dengan parameter 8 bit data, 1 stop bit, dan tidak menggunakan bit *parity*. Setelah mengatur register-register yang akan dipakai untuk komunikasi serial, untuk mengambil data dan mengirim data menggunakan fungsi sebagai berikut:

```
void USART_vSendByte(uint8_t u8Data)
{
// Wait if a byte is being transmitted
while((UCSRA&(1<<UDRE)) == 0);
// Transmit data
UDR = u8Data;
}
uint8_t USART_vReceiveByte(void)
{
// Wait until a byte has been received
while((UCSRA&(1<<RXC)) == 0);
// Return received data
return UDR;
}
```

Setelah pengaturan register-resgiter pada mikrokontroler telah selesai, kemudian atur modul *EB500* agar dapat sinkron pada saat komunikasi serial dilakukan. Pada pertama modul ini dibeli, *setting*-an berbagai parameter seperti: *baud rate*, nama *bluetooth*, dan *password* telah ditentukan oleh pabrik. Sehingga dibutuhkan pengaturan agar modul ini sesuai dengan perancangan sistem yang akan dibuat. Pada alat ini parameter yang harus diganti ialah *baudrate*, nama *bluetooth*, *password* yang akan digunakan. *Baudrate* yang akan digunakan ialah 115200 bps sesuai dengan kebutuhan protokol STK 500, nama *bluetooth* yang akan digunakan ialah *DOWNLOADER_BLUETOOTH* (pemilihan nama ini bertujuan untuk mempermudah pengguna), dan *password* yang digunakan ialah 090305. Dalam proses pengaturan menggunakan *software terminal.exe* karena komunikasi yang digunakan ialah serial. Proses pengaturan ini menggunakan *port* serial komputer.

♥ Pengaturan Baudrate

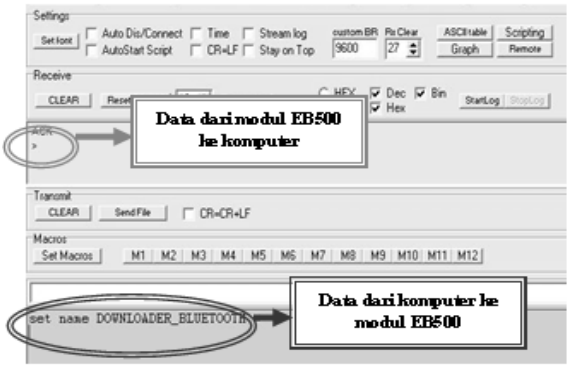
Pada *software terminal.exe* pada kolom *transmitter* (mengirim data dari komputer ke modul *EB500*), mengirimkan karakter set *baud* 115200 * [*enter*]. Setelah dikirim karakter di atas, maka modul *EB500* akan mengirim karakter *ACK* (*acknowledge*). Pada Gambar 11 disajikan tampilan *window* untuk pengaturan *baudrate*.



Gambar 11. Tampilan Pengaturan *Baudrate*

♥ Pengaturan Nama *Bluetooth*

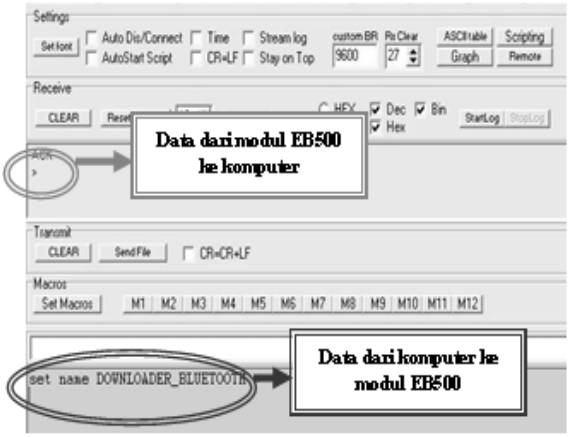
Pada *software terminal.exe* pada kolom *transmitter* (mengirim data dari komputer ke modul *EB500*), mengirimkan karakter set *name DOWNLOADER_BLUETOOTH* * [*enter*]. Setelah dikirim karakter di atas, maka modul *EB500* akan mengirim karakter *ACK* (*acknowledge*). Pada Gambar 12 disajikan tampilan *window* untuk pengaturan nama *bluetooth*.



Gambar 12. Tampilan Pengaturan Nama *Bluetooth*

♥ Pengaturan *Password*

Pada *software terminal.exe* pada kolom *transmitter* (mengirim data dari komputer ke modul *EB500*), mengirimkan karakter set *passkey* 090305 * [*enter*]. Setelah karakter di atas dikirim, maka modul *EB500* akan mengirim karakter *ACK* (*acknowledge*). Pada Gambar 13 merupakan tampilan *window* untuk pengaturan nama *bluetooth*.

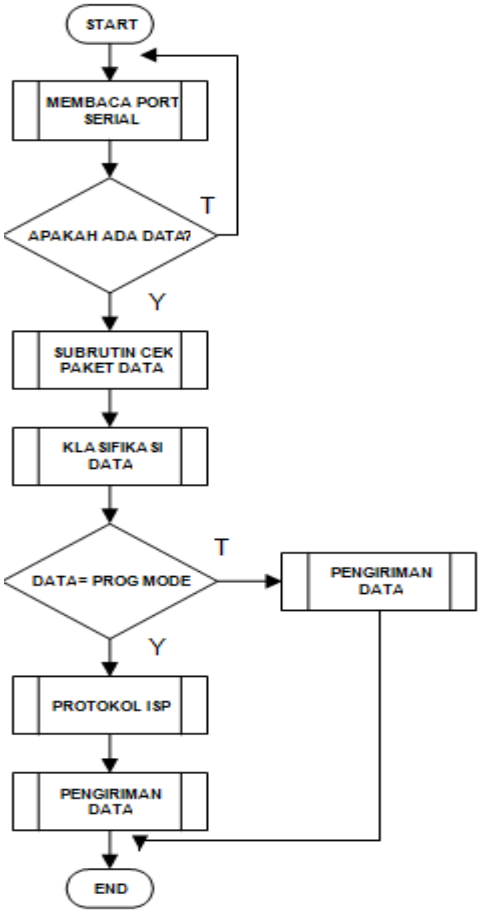


Gambar 13. Tampilan Pengaturan *Password*

Setelah pengaturan modul *EB500* selesai dilakukan selanjutnya menuju ke proses perancangan *software* mikrokontroler *ATMega8*. Mikrokontroler *ATMega8* digunakan untuk melakukan komunikasi data dengan komputer melalui *transceiver Bluetooth* dan juga untuk mengontrol mikrokontroler target yang akan di-*download*. Untuk lebih jelasnya akan dibahas diagram alir *software ATMega8* yang terbagi menjadi diagram alir utama, diagram alir membaca *port serial*, diagram alir pemisahan data, diagram alir klasifikasi data, diagram alir pengiriman ke komputer dan diagram alir protokol *ISP*.

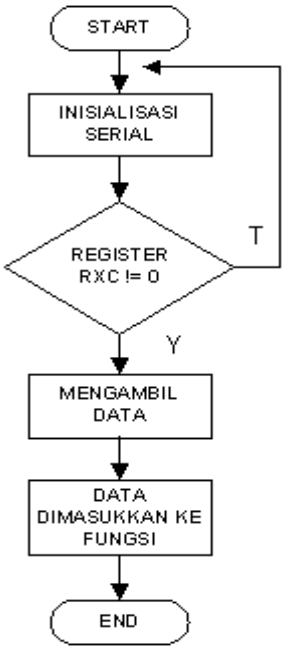
♥ Diagram Alir Utama

Pada Gambar 14 mikrokontroler ATMega8 memeriksa *port* serial apakah ada data atau tidak. Jika tidak ada data serial, maka mikrokontroler akan memeriksa kembali *port* serial. Jika ada data, maka data tersebut akan disimpan ke dalam suatu variabel. Variabel tersebut akan dipisahkan data yang ada di dalamnya, karena data tersebut mengandung *message_start*, *sequence_number*, *message_size*, *token*, *message_body*, dan *check_sum*. Setelah data dipisahkan, data *message_body* diambil dan mikrokontroler melakukan klasifikasi data tersebut. Jika data tersebut membutuhkan parameter mikrokontroler target, maka mikrokontroler akan memerintahkan mikrokontroler target masuk ke mode serial programming.



Gambar 14. Diagram Alir Utama

♥ Subrutin Membaca *Port* Serial



Gambar 15. Diagram Alir Pengambilan Data

Realisasi program pada Gambar 15. di atas adalah sebagai berikut:

```
int main(void)
{
USART_vInit();
for(;;){
stkPoll(); // fungsi untuk mengambil data
          }
          return 0;
}
//fungsi stkPoll
void    stkPoll(void)
{
          if ((UCSRA&(1<<RXC)) != 0) {
          stkSetRxChar(USART_vReceiveByte());
          }
          if(rxBlockAvailable){
rxBlockAvailable = 0;
stkEvaluateRxMessage();
}
}
```

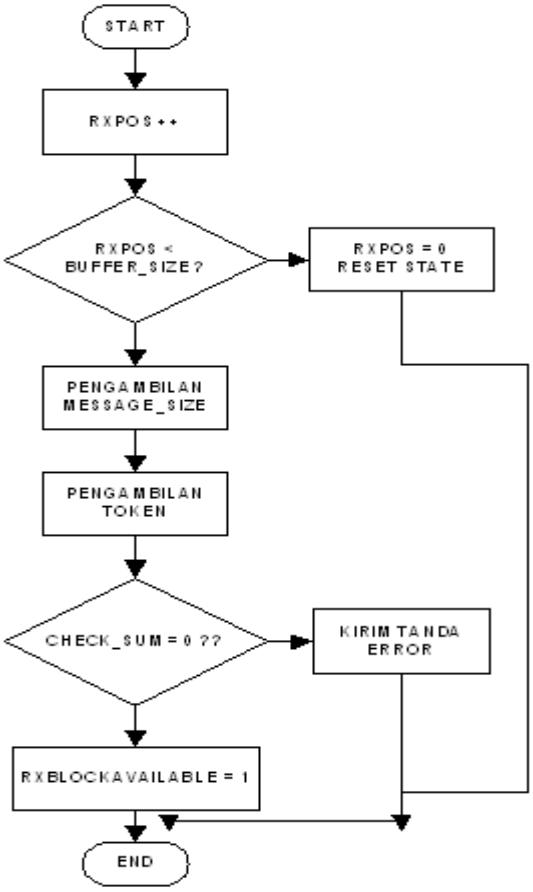
((UCSRA&(1<<RXC)) != 0) jika nilainya tidak sama dengan 0, maka memanggil fungsi menerima data dari *port* serial dan data tersebut dimasukkan ke fungsi stkSetRxChar.

♥ Subrutin Pemisahan Data

Data yang dikirimkan dari komputer mengandung banyak data yang menyusun protokol STK 500. Data tersebut terdiri dari *message_start* (1B), *sequence_number* (00 – FF), *message_size* (data 2 bit), *token* (0E), *message_body*, dan *check_sum*. Diagram alir pemrogramannya dapat dilihat pada Gambar 16.

Subrutin ini berfungsi untuk memeriksa kelengkapan paket data yang dikirimkan komputer pengguna. Keseluruhan paket data yang dikirimkan diakhiri oleh *Check_Sum*. *Check_Sum* bernilai *XOR* dari keseluruhan data yang diterima. Jika *Check_Sum* benar, maka variabel *RXBLOCKAVAILABLE* = 1 (masuk *state* selanjutnya), jika salah mikrokontroler akan mengirimkan tanda *error* ke komputer pengguna. Realisasi program sebagai berikut :

```
void stkSetRxChar(uchar c)
{
if(rxPos == 0){        /* only accept STX as the first
character */
if(c == STK_STX)
rxBuffer[rxPos++] = c; //rxpos sekarang + 1 (untuk
selanjutnya)
} //rxBuffer[] = data serial yang masuk
else{
if(rxPos < BUFFER_SIZE){
if(rxPos == 4){     /* do we have length byte? */
rxLen.bytes[0] = rxBuffer[3];
```



Gambar 16. Diagram Alir Cek Paket Data

```
rxBuffer[rxPos++] = c; //rxpos sekarang + 1 (untuk
selanjutnya)
} //rxBuffer[] = data serial yang masuk
else{
if(rxPos < BUFFER_SIZE){
if(rxPos == 4){     /* do we have length byte? */
rxLen.bytes[0] = rxBuffer[3];
rxLen.bytes[1] = rxBuffer[2];
rxLen.word += 6;
if(rxLen.word > BUFFER_SIZE){    /* illegal length */
rxPos = 0;      /* reset state */
}
}else if(rxPos == 5){
if(c != STK_TOKEN){
rxPos = 0;  /* reset state */
}
}else if(rxPos > 4 && rxPos == rxLen.word){ //
complete
uchar sum = 0;
uchar *p = rxBuffer;
while(rxPos){ /* decrement rxPos down to 0 -> reset
state */
sum ^= *p++;
rxPos--;
}
if(sum == 0){    /* check sum is correct, evaluate rx
message */
rxBlockAvailable = 1;
 }else{         /* checksum error */
txBuffer[STK_TXMSG_START]=STK_ANSWER_C
KSUM_ERROR; //kirim pos ke 5 dan 6
```

```
txBuffer[STK_TXMSG_START + 1] =STK_ANSWER_CKSUM_ERROR; //kirim pos ke 6
stkSetTxMessage(2);
}
}
}else{  /* overflow */
 rxPos = 0;  /* reset state */
}
}
```

♥ **Subrutin Klasifikasi Data**

Perintah yang dikirimkan oleh komputer pengguna harus direspon oleh mikrokontroler. Oleh karena itu perintah tersebut harus di klasifikasikan terlebih dahulu. Perintah-perintah tersebut dibaca oleh mikrokontroler, sehingga mikrokontroler dapat menentukan respon atas perintah yang dikirimkan oleh komputer. Realisasi program sebagai berikut:

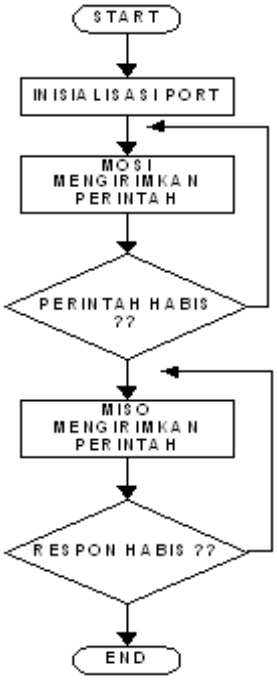
```
void stkEvaluateRxMessage(void) /* not static to prevent inlining */
{
uchar      i, cmd;
utilWord_t  len = {2};  /* defaults to cmd + error code */
void *param;
          cmd = rxBuffer[STK_TXMSG_START];
txBuffer[STK_TXMSG_START] = cmd;
txBuffer[STK_TXMSG_START+1]= STK_STATUS_CMD_OK;
param = &rxBuffer[STK_TXMSG_START + 1];
          if(cmd==STK_CMD_SIGN_ON){
static PROGMEM uchar string[] = {8, 'S', 'T', 'K', '5', '0', '0', '_', '2', 0};
uchar *p = &txBuffer[STK_TXMSG_START + 2];
strcpy_P(p, string);
len.bytes[0] = 11;
}
          else if(cmd==STK_CMD_SET_PARAMETER){
setParameter(rxBuffer[STK_TXMSG_START+1], rxBuffer[STK_TXMSG_START + 2]);
          }
else if(cmd==STK_CMD_GET_PARAMETER){
txBuffer[STK_TXMSG_START+2]=getParameter(rxBuffer[STK_TXMSG_START + 1]);
len.bytes[0] = 3;
}
else if(cmd==STK_CMD_OSCCAL){
txBuffer[STK_TXMSG_START + 1] = STK_STATUS_CMD_FAILED;
}
          else if(cmd==STK_CMD_LOAD_ADDRESS){
for(i=0;i<4;i++){
stkAddress.bytes[3-i] = rxBuffer[STK_TXMSG_START + 1 + i];
}
          }
          else if(cmd==STK_CMD_ENTER_PROGMODE_ISP){
txBuffer[STK_TXMSG_START + 1] = ispEnterProgmode(param);
}
else if(cmd==STK_CMD_LEAVE_PROGMODE_ISP){
ispLeaveProgmode(param);
}
else if(cmd==STK_CMD_CHIP_ERASE_ISP){
txBuffer[STK_TXMSG_START + 1] = ispChipErase(param);
}
else if(cmd==STK_CMD_PROGRAM_FLASH_ISP){
txBuffer[STK_TXMSG_START + 1] = ispProgramMemory(param, 0);
}
          else if(cmd==STK_CMD_READ_FLASH_ISP){
len.word=1+ispReadMemory(param,(void*)&txBuffer[STK_TXMSG_START + 1], 0);
}
          else if(cmd==STK_CMD_PROGRAM_EEPROM_ISP){
txBuffer[STK_TXMSG_START + 1] = ispProgramMemory(param, 1);
}
          else if(cmd==STK_CMD_READ_EEPROM_ISP){
len.word=1+ispReadMemory(param,(void*)&txBuffer[STK_TXMSG_START + 1], 1);
}
          else if(cmd==STK_CMD_PROGRAM_FUSE_ISP){
txBuffer[STK_TXMSG_START + 1] = ispProgramFuse(param);
}
          else if((cmd==STK_CMD_READ_FUSE_ISP)||(cmd==STK_CMD_READ_LOCK_ISP)||(cmd==STK_CMD_READ_SIGNATURE_ISP)||(cmd==STK_CMD_READ_OSCCAL_ISP)){
txBuffer[STK_TXMSG_START + 2] = ispReadFuse(param);
txBuffer[STK_TXMSG_START + 3] = STK_STATUS_CMD_OK;
len.bytes[0] = 4;
}
          else if(cmd==STK_CMD_PROGRAM_LOCK_ISP){
txBuffer[STK_TXMSG_START + 1] = ispProgramFuse(param);
}
          else if(cmd==STK_CMD_SPI_MULTI){
len.word= 1 + ispMulti(param, (void *)&txBuffer[STK_TXMSG_START + 1]);
}
```

Dari potongan program di atas, mikrokontroler membaca perintah yang dikirimkan dan merespon atas perintah dari komputer sesuai ketentuan protokol STK500. Misal, perintah yang dikirimkan ialah *STK_CMD_SIGN_ON*, maka mikrokontroler mengirimkan karakter STK500_2 menuju komputer melalui *port* serial.

♥ **Subrutin Protokol *ISP***

Subrutin ini berfungsi untuk mengontrol mikrokontroler target agar mikrokontroler target masuk pada mode serial *programming*. Setelah masuk ke serial *programming*, maka mikrokontroler *downloader* akan bertindak sebagai master dan mikrokontroler target akan bertindak menjadi *slave*.

Diagram alir pemrogramannya dapat dilihat pada Gambar 17.



Gambar 17. Diagram Alir Protokol *ISP*

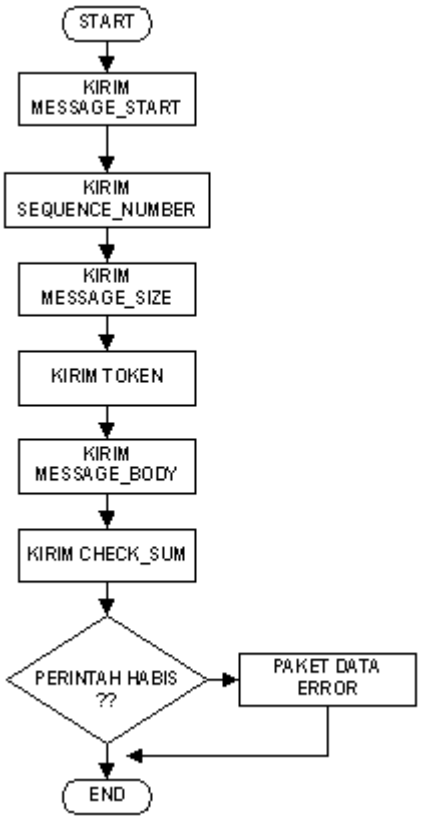
Realisasi program sebagai berikut:

```
static uchar ispBlockTransfer(uchar *block, uchar len)
{
uchar   cnt, shift = 0, port, delay = ispClockDelay;
cli();
            port=PORT_OUT(HWPIN_ISP_MOSI)&~(1
<<PORT_BIT(HWPIN_ISP_MOSI));
            while(len--){   /* len may be 0 */
port &= ~(1 << PORT_BIT(HWPIN_ISP_MOSI));
if(PORT_PIN_VALUE(HWPIN_ISP_MISO))
shift |= 1;
PORT_PIN_CLR(HWPIN_ISP_SCK);     // <-- device
changes data
}while(--cnt);
            }
return shift;
}
```

Port MOSI akan mengirim suatu parameter sesuai perintah yang dikirimkan komputer sampai data habis, setelah itu *port MISO* akan menerima data hasil respon perintah yang dikirimkan. Data yang dikirimkan sesuai aturan protokol *In System Programming.*

♥ **Subrutin Kirim Data Ke Komputer**

Fungsi ini digunakan untuk mengirim data ke komputer melalui *port* serial mikrokontroler. Pada protokol STK 500 mengharuskan mikrokontroler *downloader* mengirim respon perintah yang diterima dari komputer. Diagram alir pemrogramannya dapat dilihat pada Gambar 18.



Gambar 18. Diagram Alir Pengiriman Data Ke Komputer

Implementasi ke dalam program sebagai berikut:

```
static void stkSetTxMessage(uint len)
{
uchar   *p = txBuffer, sum = 0;
            *p++ = STK_STX;             //message  start
*p++ = rxBuffer[1];  /* sequence number */
*p++ = utilHi8(len); //message size msb
*p++ = len;          //message size lsb
*p++ = STK_TOKEN; //token
txPos = 0;
len += 6;
txLen = len--;
p = txBuffer;
while(len--){
sum ^= *p++;}
*p = sum;
            for (a=0;a<txLen;a++) {
```

```
        USART_vSendByte(txBuffer[a]);
        }
}
```

USART_vSendByte(txBuffer[a]); ialah fungsi untuk mengirimkan data ke komputer dari *port* serial mikrokontroler *downloader*.

Alat ini digunakan untuk men-*download* program ke mikrokontroler target dengan menggunakan koneksi *bluetooth*. Prosedur yang dilakukan pengguna sebagai berikut:

1. Komputer atau laptop yang akan digunakan telah terpasang *bluetooth dongle* dan *driver*-nya telah terinstal;
2. Catu daya mikrokontroler target yang akan di program dinyalakan;
3. Pasangkan alat ini ke mikrokontroler target;
4. Koneksikan *bluetooth dongle* dengan alat ini, setelah terkoneksi, maka alat ini akan memberikan alamat *com port* untuk jalur komunikasinya;
5. Pada *software compiler CodeVisionAVR* atur penggunaan *com port* dan protokol yang akan digunakan. Atur protokol yang digunakan ialah *Atmel STK500/AVRISP* dan *com port* yang digunakan harus sesuai dengan *com port* yang telah diberikan oleh modul *EB500*;
6. Untuk meng-*compile* dan men-*download* program tekan *shift + F9.*

HASIL PENELITIAN DAN PEMBAHASAN

Pada bagian selanjutnya akan dilakukan pengukuran dan pengujian alat yang meliputi pengujian modul *Bluetooth EB500*, pembuktian program, pengujian waktu yang dibutuhkan untuk men-*download* program dengan jarak sebagai variabel pembandingnya, pengujian waktu yang dibutuhkan untuk men-*download* program dengan besar program sebagai variabel pembandingnya, dan pengukuran dan pengujian alat secara keseluruhan. Hasil dari pengujian diharapkan dapat mengetahui seberapa besar tingkat keberhasilan dan kesalahannya. Alat yang digunakan untuk melakukan pengujian meliputi:

- *Stopwatch*;
- *Software Free Port Serial Monitor*;
- *Software* Terminal;
- Meteran.

Pengujian dilakukan terhadap Modul *Bluetooth EB500* dengan menggunakan *software terminal.exe* yang sering dipakai untuk mengecek konektivitas *Port Serial*. Adapun hubungan antara *PC*, *EB500* serta mikrokontroler dapat dilihat pada Gambar 19 di bawah ini. Pengukuran dilakukan dengan mengirimkan beragam karakter kode dari *PC* yang telah ditentukan sebelumnya serta melihat karakter yang dikirimkan kembali ke *PC* oleh mikrokontroler melalui Modul *Bluetooth EB500* tersebut dalam jarak 5 m, jarak tersebut dipilih karena ruangan presentasi tempat alat tersebut nantinya berada tidak terlalu besar, sehingga jarak antara pengguna dan alat tersebut juga tidak telalu jauh. Komunikasi *Bluetooth* yang akan diuji ini menggunakan *Port* komunikasi *COM4*.

Gambar 19. Hasil Pengujian Sinkronisasi Antara Modul *EB500* Dengan *Bluetooth Dongle*

Dari hasil pengujian sinkronisasi modul *EB500* dengan komputer dapat dilihat pada Gambar 19**.** Data yang dikirimkan dari komputer dapat diterima dan dikirimkan kembali ke komputer sesuai dengan data yang di-*input*-kan oleh komputer. Hal tersebut dapat tercapai karena modul *EB500* dan komputer pengguna telah tersinkronisasi dengan baik. Karena parameter antara lain: *baudrate*, ukuran data, *stop* bit kedua *device* ini telah sama.

Aplikasi dari sinkronisasi antara komputer dan modul ini terletak pada respon alat ini atas data yang dikirimkan oleh komputer pengguna. Jika tidak sinkron, maka alat ini tidak akan merespon *request* dari komputer pengguna, karena data yang diterima oleh alat ini tidak sesuai dengan data yang dikirimkan oleh komputer pengguna.

Pembuktian program yang digunakan alat ini dengan menggunakan *software Free Port Serial Monitor* yang berfungsi untuk memonitor *port serial* dari komputer. Pada sisi pengguna terdapat komputer yang telah terpasang *bluetooth dongle*, dan terdapat didalamnya *software compiler CodeVisionAVR*. Sebelum menggunakan alat ini, pada *software compiler*

di-*setting* terlebih dahulu jenis *programmer* yang digunakan dan pada *com* yang telah ditentukan. Pada pembuktian program yang dilakukan menggunakan perintah *read_chip_signature*, yaitu perintah untuk membaca jenis *chip* pada mikrokontroler target. Pada *compiler CodeVisionAVR* pilih *tools→Chip Programmer→ read→ Chip Signature.*

Dari hasil pembuktian di atas dapat dilihat bahwa protokol yang digunakan telah sesuai dengan dasar teori protokol STK 500. Pembuktian di atas dibagi menjadi 2 bagian, yaitu dari sisi mikrokontroler dan sisi komputer. Pada sisi komputer terlihat bahwa setiap melakukan perintah (misal: *CMD_SIGN_ON*) selalu dibungkus oleh suatu aturan pengiriman. Pada sisi mikrokontroler terdapat data 1B 01 00 0B 0E 01 08 53 54 4B 35 30 30 5F 32 02. Data tersebut ialah pengiriman respon mikrokontroler terhadap perintah dari komputer dengan perintah *CMD_SIGN_ON* dengan cara pengiriman karakter STK500_2. Pengiriman data tersebut selalu dibungkus dengan protokol STK 500 dengan melalui *MESSAGE_START (0x1B), SEQUENCE_NUMBER (0x00 – 0xFF), MESSAGE_SIZE (*data 2 bit*), TOKEN (0x0E), MESSAGE_BODY (*0 sampai 65535 bit*), dan CHECK_SUM (*data 1 bit*).*

Pengujian waktu yang dibutuhkan untuk proses *download* alat ini menggunakan *software compiler CodeVisionAVR* dengan besar memory *flash* 7,2 % dari *memory flash* dari mikrokontroler *ATMega32*. Pengukuran ini menggunakan bantuan *stopwatch digital* untuk mengukur waktu yang dibutuhkan untuk men-*download* program sebesar 7,2 % dari memori *flash ATMega32*. Setelah program telah dituliskan, tekan *shift + F9* untuk memulai men-*download* program ke mikrokontroler target. Bersamaan dengan *stopwatch* memulai berjalan sampai program telah selesai di-*download*. Hasil dari pengukuran dapat dilihat pada Tabel 2. Prosedur pengukuran dilakukan berulang-ulang dengan mengubah jarak antara pengguna dan *downloader*.

Dari pengukuran waktu yang dibutuhkan proses *download* dengan metode *line of sight* (tanpa halangan) terlihat bahwa pada jarak 17 meter *bluetooth* tidak dapat terdeteksi oleh pengguna. *Area coverage* dari modul *EB500* adalah 10 meter karena pada modul ini kelas *bluetooth* yang terpasang ialah kelas 2. *Area coverage* modul *EB500* terlihat bahwa modul ini mampu mencakup area sampai dengan jarak 16 meter. Waktu yang dibutuhkan untuk *download* program dengan ukuran 7,2 % dari

Tabel 2. Hasil Pengukuran Waktu Yang Dibutuhkan Proses *Download* Mikrokontroler Target Dengan Metode *Line of Sight*

Jarak (m)	Waktu (detik)
1	14,53
2	16,78
3	16,13
4	15,9
5	14,53
6	17,66
7	17,28
8	17,97
9	16,67
10	16,16
11	17,01
12	16,48
13	15,96
14	17,26
15	19,24
16	17,11
17	Tidak Terdeteksi
18	Tidak Terdeteksi
19	Tidak Terdeteksi
20	Tidak Terdeteksi

memory flash dari mikrokontroler *ATMega32* membutuhkan waktu 14-20 detik. Dari pengukuran di atas terlihat bahwa waktu yang dibutuhkan tidak konstan selalu berubah-ubah, hal ini dikarenakan media perantara berupa udara yang mengandung banyak partikel-partikel, sehingga komunikasi antara pengguna dan alat sedikit terganggu.

Setelah program telah dituliskan, tekan shift + F9 untuk memulai men-*download* program ke mikrokontroler target. Bersamaan dengan *stopwatch* memulai berjalan sampai program telah selesai di *download*. Hasil dari pengukuran dapat dilihat pada Tabel 3. Prosedur pengukuran dilakukan berulang-ulang dengan mengubah jarak antara pengguna dan *downloader*.

Penghalang yang terpasang pada alat dipasang secara keseluruhan menutupi alat ini, melainkan masih terdapat celah udara yang masuk. Material penghalang antara lain: triplek, alumunium, plastik, dan kardus. Pada pengujian waktu yang dibutuhkan untuk proses *download* dengan menggunakan penghalang dengan berbagai macam material terlihat bahwa penghalang dapat mempengaruhi waktu *download*. Waktu yang dibutuhkan sangat bervariatif karena media perantara komunikasi menggunakan udara yang mengandung banyak partikel-partikel, sehingga komunikasinya terganggu. Dan juga dengan adanya penghalang, maka akan semakin menambah lama waktu *download*.

Tabel 3. Hasil Pengukuran Kecepatan Proses Download Mikrokontroler Target Dengan Menggunakan Halangan

Jarak (m)	Waktu dengan halangan triplek (detik)	Waktu dengan halangan alumunium (detik)	Waktu dengan halangan plastik (detik)	Waktu dengan halangan kardus (detik)
1	14,37	16,96	16,32	14,47
2	15,81	16,18	16,34	14,97
3	16,59	16,28	16,29	16,22
4	16,82	15,47	16,06	14,82
5	14,25	16,78	14	14,94
6	14,03	22,28	15,16	14,6
7	14,28	25,72	19,91	16,22
8	19,29	18,45	19,03	15,67
9	18,87	21,23	15,28	15,11
10	19,74	21,25	16,86	16,22
11	18,6	Tidak Terdeteksi	15,75	16,68
12	18,77	Tidak Terdeteksi	19,59	17,49
13	17,31	Tidak Terdeteksi	22,51	16,92
14	19,26	Tidak Terdeteksi	16,15	19,02
15	16,2	Tidak Terdeteksi	16,36	19,29
16	16,26	Tidak Terdeteksi	37,34	45,51
17	Tidak Terdeteksi	Tidak Terdeteksi	Tidak Terdeteksi	Tidak Terdeteksi
18	Tidak Terdeteksi	Tidak Terdeteksi	Tidak Terdeteksi	Tidak Terdeteksi

Pengujian waktu yang dibutuhkan untuk proses *download* alat dengan ukuran program sebagai variabel pembandingnya menggunakan *software compiler CodeVisionAVR* dan *stopwatch*. Setelah program telah dituliskan, tekan *shift + F9* untuk memulai men-*download* program ke mikrokontroler target. Bersamaan dengan *stopwatch* memulai berjalan sampai program telah selesai di *download*. Hasil dari pengukuran dapat dilihat pada Tabel 4. Prosedur pengukuran dilakukan berulang-ulang dengan mengubah jarak antara pengguna dan *downloader*.

Dari hasil pengujian di atas terlihat bahwa besar ukuran program yang akan di-*download* ke mikrokontroler target sangat berpengaruh terhadap waktu yang dibutuhkan untuk proses *download* dari alat ini. Semakin besar ukuran program yang akan di-*downlaod*, maka waktu *download* akan semakin lama, dan sebaliknya.

Tabel 4. Hasil Pengukuran Kecepatan Proses Download Mikrokontroler Target

Jarak (m)	Ukuran (bit)	Waktu (detik)
5	606	7,73
5	1690	12,54
5	2370	15,33
5	3372	19,23
5	3918	21,47
5	4532	23,25
5	4918	25,33
5	5402	28,81
5	5896	30,79
5	6574	32,78
5	7034	34,36

Hal ini dikarenakan karena semakin besar program maka semakin banyak data yang harus dikirimkan ke mikrokontroler target, sehingga dibutuhkan waktu yang panjang untuk ukuran program yang besar. Alat ini bertujuan untuk mengurangi penggunaan tenaga bagi *programmer* untuk memprogram suatu mikrokontroler yang akan diprogram khususnya mikrokontroler bergerak. Pada Tabel 5 disajikan perbedaan penggunaan tenaga *programmer* antara *downloader* kabel dan *downloader bluetooth.*

Tabel 5. Tabel Perbedaan Penggunaa Tenaga Antara *Downloader* Kabel dan *Downloader Bluetooth*

No.	*Downloade*r Kabel	*Downloader Bluetooth*
1	Jika ingin men-*downloa*d ulang robot, robot harus angkat dan didekatkan ke komputer pengguna.	Jika ingin men-*download* ulang robot, robot tidak perlu di angkat mendekati komputer pengguna.
2	Kabel *down-loader* (*USB*) harus di koneksikan ke komputer pengguna setiap kali *programmer* ingin men-*download* ulang.	Tidak perlu mengkoneksikan kabel apapun ke komputer pengguna, karena menggunakan koneksi *bluetooth*.

Dari Tabel 5 di atas dapat diambil kesimpulan bahwa dengan menggunakan alat ini *(downloader bluetooth)* para pengguna dapat menghemat penggunaan tenaga mereka, karena tidak perlu lagi mengangkat kembali mikrokontroler bergerak (*robot*) mendekati pengguna.

Manfaat alat ini sangat besar, karena mampu mengurangi penggunaan tenaga *programmer* secara signifikan. Untuk menggunakan alat ini para pengguna harus

membeli alat ini sekitar Rp. 500.000,00. Daftar komponen yang menyusun alat ini dapat dilihat pada Tabel 6.

Tabel 6. Daftar Komponen Alat

No.	Komponen	Harga, Rp.
1	Rangkaian Minimum Sistem *ATMega8*	70.000
2	Modul *EB 500*	300.000
3	Kabel + *Casing*	30.000
4	*Bluetooth Dongl*e	20.000
5	CD + Kertas + Plastik	5.000
	Total	420.000

Pembeli yang akan membeli alat ini akan mendapat 1 paket pembelian yang terdiri dari *downloader AVR bluetooth, CD + Manual book, bluetooth dongle,* dan kemasan berupa plastik. Harga jual Rp. 500.000,00 dan harga produksi sebesar Rp. 420.000,00, jika laku terjual maka akan mendapatkan keuntungan Rp. 80.000,00. Pangsa penjualan alat ini ialah toko elektronik (*digi-ware, delta-electronic, el-tech*, dan lain-lainnya) yang akan membeli partai (grosir), dan publik terkait.

Pada pengukuran alat ini akan diuji dengan men-*download* program menuju suatu robot. Pengguna akan men-*download* program dengan berbagai algoritma jalan yang berbeda. Setelah program telah dituliskan, tekan *shift + F9* untuk memulai men-*download* program ke mikrokontroler target. Bersamaan dengan *stopwatch* memulai berjalan sampai program telah selesai di-*download*. Mikrokontroler target yang berada pada badan robot bertipe *AT Mega32*. Hasil dari pengukuran dapat dilihat pada Tabel 7. Prosedur pengukuran dilakukan berulang-ulang dengan mengubah algoritma jalan robot.

Tabel 7. Hasil Pengujian Alat Dengan Menggunakan Robot Sebagai Mikrokontroler Target

Algoritma jalan robot	Keterangan
Maju sampai ada halangan didepan robot dan berhenti	Sesuai
Maju sampai ada halangan didepan robot dan hadap kanan	Sesuai
Maju sampai ada halangan didepan robot dan hadap kiri	Sesuai
Maju sampai sisi kanan robot kosong dan berhenti	Sesuai
Maju sampai sisi kiri robot kosong dan berhenti	Sesuai

Dari hasil pengujian di atas, dapat disimpulkan bahwa program yang di-*download* oleh pengguna sesuai dengan program yang telah direncanakan.

KESIMPULAN

Dari hasil penelitian dan pembahasan yang meliputi hasil perancangan, pembuatan dan pengujian alat, maka dapat ditarik beberapa kesimpulan antara lain:

1. Alat ini dapat menghemat tenaga *programmer* dengan sangat signifikan;
2. Alat ini memiliki nilai jual yang tinggi karena memiliki manfaat yang tinggi, harga jual alat ini Rp. 500.000,00 dengan biaya produksi Rp. 420.000,00 sehingga keuntungan sebesar Rp. 80.000,00;
3. Jangkauan *area coverage* dari modul *EB500* ini sampai 10 meter, tetapi setelah dilakukan pengujian *area coverage* menjadi 16 meter;
4. Waktu yang dibutuhkan dengan menggunakan metode *line of sight* (tanpa halangan) dengan jarak antara pengguna dan alat antara 1-16 meter dan besar program 7,2 % dari memori *flash* mikrokontroler AT Mega 32 antara 14 – 20 detik.
5. Waktu yang dibutuhkan dengan menggunakan halangan berbagai material dengan jarak
6. antara pengguna dan alat antara 1 – 16 meter besar program 7,2 % dari *memory flash* mikrokontroler *ATMega32* berkisar antara 14-45 detik;
7. Waktu yang dibutuhkan dengan menggunakan metode *line of sight* pada jarak 5 meter dengan ukuran program antara 606 sampai 7034, sebesar 7-34 detik.

DAFTAR PUSTAKA

[1] Mazidi, M. A., dan Kawan-kawan, *The 8051 Microcontroller and Embedded Systems Using Assembly and C*, Hlm. 28, Pearson Education, New Jersey, 2006

[2] Atmel Corporation, *ATMega8 Datasheet*, *Atmel 8-bit AVR Microcontroller with 8K Bytes in-system Programmable Flash*, Atmel Corporation, San Jose:2008

[3] Court, D., *EB500*, www.parallax.com/dl/docs/prod/comm/eb500.pdf., Diakses 5 Januari 2011

[4] Atmel Corporation, *STK500 Communication Protocol*, www.atmel.com/dyn/resources/prod_documents/doc2591.pdf., Diakses 5 Januari 2011

[5] Atmel Corporation, *In-System Programming*, ww.atmel.com/dyn/resources/prod_documents/doc0943.pdf, Diakses 5 Januari 2011