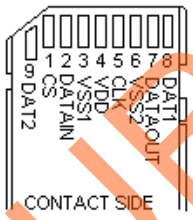
BAB IV

HASIL KERJA PRAKTEK

4.1 Akses MMC dengan Microcontroller

MMC (Multi Media Card) merupakan alat untuk menyimpan data digital. *Memory card* biasanya mempunyai kapasitas ukuran berdasarkan bit digital, yaitu 16 MB, 32 MB, dan seterusnya. Bisa menggunakan *memory card* tipe apapun, yang penting harus menambahkan dengan *adapter* dengan kurang lebih seperti di bawah ini.



Gambar 4.1 Kofigurasi MMC

Konfigurasi Pin MMC

Pin 1 = CS = chip select

Pin 2 = DI = data input

Pin 3 = Vss =ground

Pin 4 = Vcc ,tegangan 3,3Volt.

Pin 5 = SCLK = serial clock

Pin 6 = Vss2 = ground

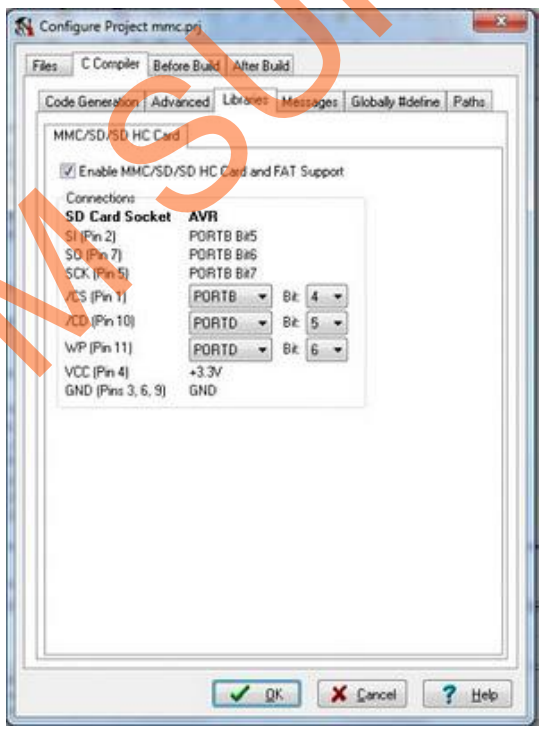
Pin 7 = DO = data out

Pin 8 = DAT1

Pin 9 = DAT2

Untuk bisa mengakses MMC menggunakan mikrokontroler,

1. Memiliki RAM 2KB atau lebih, bisa menggunakan ATMega32.
2. Menggunakan program Codevision versi 2.04 ke atas, karena adanya library MMC.
3. Codevision 2.05 telah mendukung library untuk MMC, sehingga akan lebih mudah kita untuk mengakses MMC. Pada dasarnya Codevision 2.04 telah memiliki library ini, hanya saja pada saat artikel ini dibuat versi terbaru dari codevision versi 2.05.
4. Berikut pengaturan yang dilakukan pada Codevision untuk menggunakan library MMC.



Gambar 4.2 Konfigurasi library untuk MMC pada Code Vision AVR

Untuk PORTD.5 dan PORTD.6 disambungkan ke ground saja, karena pada pin MMC tidak terdapat pin /CD dan WP.

4.2 Mengakses MMC

4.2.1 Memberi nama file

Memberi nama file tidak boleh lebih dari 8 huruf, pada contoh program yang disertakan, menggunakan nama "suhu.txt". Nama file ini hanya dihitung 4 huruf saja.

4.2.2 Membuat file

Urutan dalam membuat file adalah sebagai berikut :

- reset nama file
- beri nama file
- create file
- jangan lupa close file (kalo fungsi close file tidak di panggil maka proses membuat file baru tidak akan berhasil.

4.2.3 Mengisi File

Urutan dalam mengisi file dalah sebagai berikut :

- reset nama file
- tunjuk nama file
- baca ukuran file
- buka file kemudian pilih mode tulis

- tunjuk alamat file yang akan ditulis
- tulis file dari buffer yang telah disiapkan

jangan lupa close file (kalo fungsi close file tidak di panggil mas proses membuat file baru tidak akan berhasil.

4.3 Pembuatan Program

Program komputer (juga disebut sebagai software atau hanya program) adalah serangkaian instruksi berurutan yang ditulis untuk melakukan serangkaian tugas untuk komputer. Komputer tak memiliki kemampuan untuk menyelesaikan masalahnya sendiri. Komputer hanyalah berupa alat yang digunakan untuk melakukan perhitungan-perhitungan dan serangkaian tugas yang dibebankan kepadanya. Pembuat program disebut sebagai programmer.

secara umum, program memberikan kemampuan komputer untuk melakukan fungsi khusus. Komputer akan mengeksekusi atau menjalankan instruksi yang ada dalam program di dalam central processor. Program dibuat dengan menggunakan **bahasa pemrograman** sebagai alat untuk mengungkapkan ide sang programmer agar dapat dijalankan oleh komputer. Setelah ditulis dalam bahasa pemprograman, tidak serta merta apa yang kita tulis tersebut dapat dilaksanakan oleh komputer. Komputer hanya mengerti bilangan biner dan instruksi-instruksi menggunakan bahasa mesin. Untuk membuat komputer mengerti maksud dan tujuan kita, diperlukanlah sebuah kompiler yang dapat menerjemahkan bahasa pemprograman ke bahasa mesin. Namun ada juga komputer program yang tidak berupa kode bahasa mesin, namun berupa kode-kode khusus dan

terkadang juga berupa kode-kode bahasa pemprograman. Program model ini memerlukan bantuan interpreter untuk menerjemahkan bahasa tersebut ke bahasa mesin sehingga dimengerti oleh perangkat keras.

Source code ditulis oleh seorang programmer dan ditulis dengan salah satu bahasa pemprograman menggunakan dua paradigma utama yaitu pemprograman **imperatif** atau pemprograman **deklaratif**.

Disini Penulis menggunakan CodeVision AVR sebagai aplikasi untuk memprogram microcontroller ATMega32.

4.3.1 Pengenalan CodeVision AVR

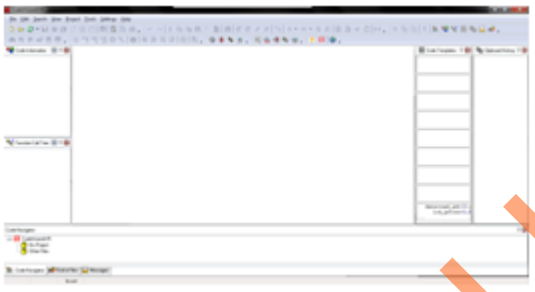
CodeVision AVR Penggunaan mikrokontroler sekarang ini telah umum. Mulai dari penggunaan untuk kontrol sederhana sampai kontrol yang cukup kompleks, mikrokontroler dapat berfungsi jika telah diisi sebuah program, pengisian program ini dapat dilakukan menggunakan compiler yang selanjutnya didownload ke dalam mikrokontroler menggunakan *downloader*. Salah satu compiler program yang umum digunakan sekarang ini adalah CodeVision AVR yang menggunakan bahasa pemrograman C.

CodeVision AVR mempunyai suatu keunggulan dari compiler lain, yaitu adanya codewizard, fasilitas ini memudahkan kita dalam inisialisasi mikrokontroler yang akan kita gunakan, codevision telah menyediakan konfigurasi yang bisa diatur pada masing-masing chip mikrokontroler yang akan kita gunakan, sehingga kita tidak perlu melihat *datasheet* untuk sekedar mengonfigurasi mikrokontroler. Berikut ini langkah-langkah menggunakan codevision.

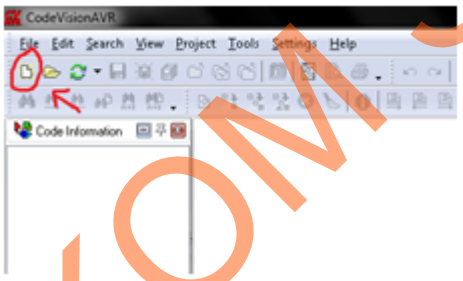
4.3.2 Cara Pengoperasian Aplikasi

1. Buka aplikasi CodeVision AVR

2. Setelah itu jendela CodeVision akan terbuka, dan menampilkan project kosong.



3. Pilih Create a New File or Project

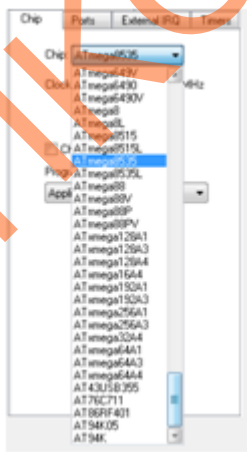


4. Setelah itu muncul dialog box untuk memilih tipe file yang akan dibuat. Pilih file project, kemudian tekan OK.

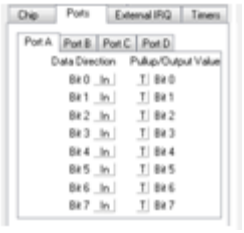
5. Ketika dikonfirmasi apakah ingin menggunakan CodeWizard?pilih Yes. Ini merupakan fasilitas yang dapat memudahkan pemrogram melakukan konfigurasi mikrokontroler.

6. Jendela baru CodeWizard akan muncul seperti gambar berikut :

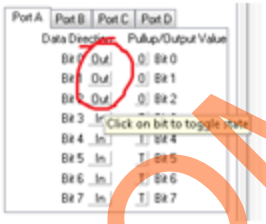
7. Tab Chip merupakan konfigurasi tipe mikrokontroler yang akan digunakan dan frekuensi yang akan diterapkan pada mikrokontroler. Jika kita skrol pada pilihan Chip, kita dapat memilih berbagai tipe mikrokontroler yang dapat digunakan.



8. Jika kita memilih Atmega32, kita dapat menggunakan frekuensi sebesar 12MHz untuk mendapatkan kinerja optimal.
9. Selanjutnya beralih ke tab Ports, tab ini digunakan untuk konfigurasi Pin input dan output pada mikrokontroler, apakah kita ingin menggunakan beberapa sebagai input dan sisanya sebagai output, kita dapat mengonfigurasinya melalui tab Ports.



10. Pada gambar diatas ditunjukkan bahwa Port A seluruhnya digunakan sebagai input. Jika kita ingin mengganti beberapa pin sebagai output, maka klik tombol disebelah tulisan Bit ‘x’. Seperti dicontohkan pada gambar berikut :



11. Perlakuan yang sama dapat diterapkan untuk Port B, Port C, dan Port D.
12. Kemudian Pilih File -> Generate, Save, and Exit, setelah itu tampil dialog box untuk menyimpan File source (*.c), file project (*.prj), dan file CodeVisionAVR project (*.cwp). Simpan dengan nama yang sama untuk memudahkan pengelompokan file.
13. Setelah itu muncul source code dengan konfigurasi mikro yang sesuai dengan pemilihan pada wizard.

Pembuatan program selesai, maka wajib untuk meng-load ke dalam microcontroller yang sudah terbuhung dengan minimum sistem yang dapat membantu untuk memasukkan program dalam chip microcontroller tersebut.

4.4 Tahap Penggabungan Antar Rangkaian Hasil Kerja

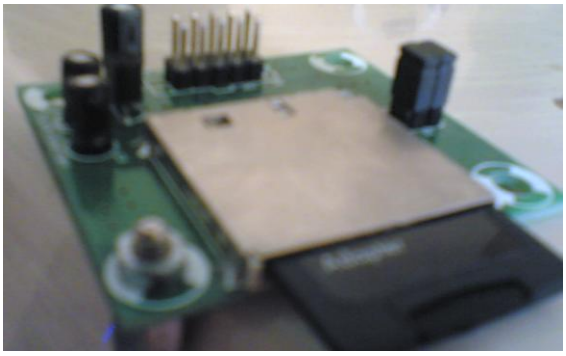
Setelah kita membuat seluruh rangkaian yang diperlukan seperti pada tahapan perancangan rangkaian elektronika di atas, maka sekarang adalah tahapan untuk menggabungkan seluruh elemen yang penulis buat untuk di gabungkan menjadi satu rangkaian.

Tahapan yang perlu untuk digabungkan adalah sebagai berikut :

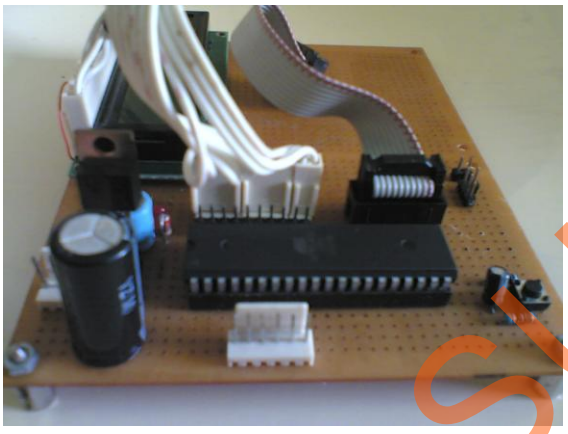
1. Rangkaian module MMC
2. Rangkaian microcontroller ATMega32
3. Peletakkan limit LCD sebagai keluaran data yang akan di tampilkan

4.4.1 Foto Alat Yang Telah Fix

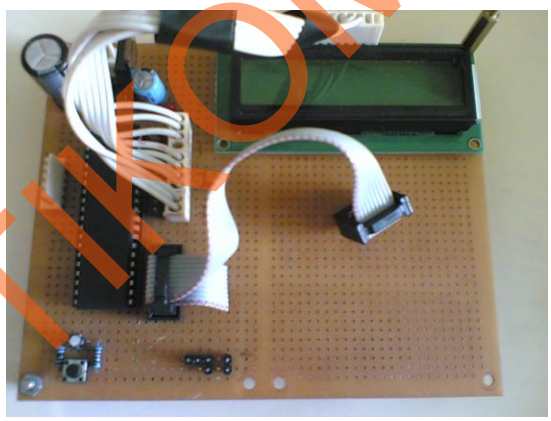
Gambar 5.1 Module MMC



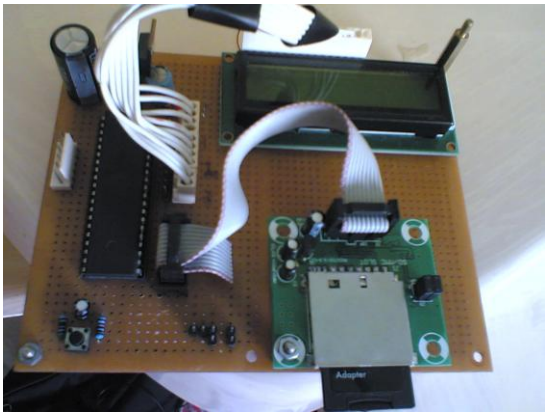
Gambar 5.2 MMC yang sudah di colokkan ke Module



Gambar 5.3 Minimun sistem Atmega 32

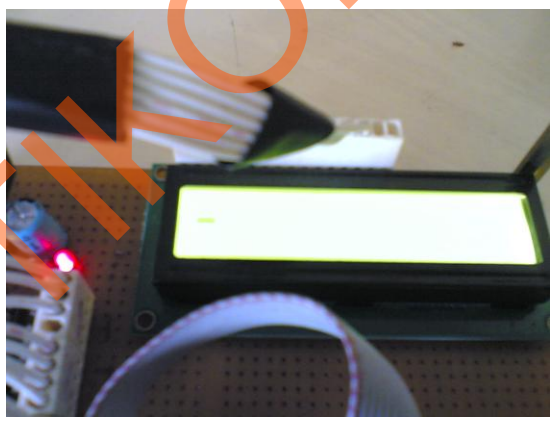


Gambar 5.4 Tampak LCD dan minimun sistem

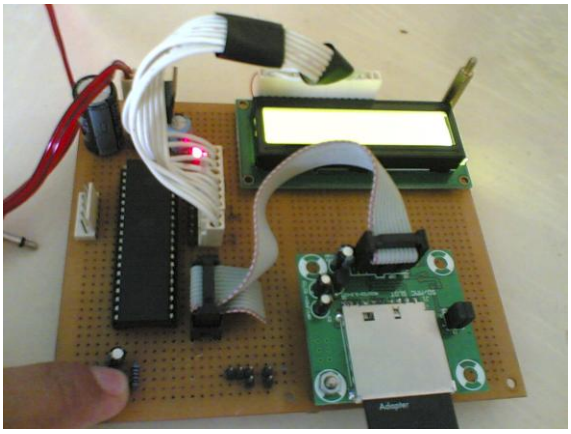


Gambar 5.5 Alat telah fix antara Module MMC, Atmega 32, dan LCD

Gambar 5.6 Adaptor sebagai suplay tegangan



Gambar 5.7 Alat telah memproses data dari MMC tampak pada layar LCD



Gambar 5.8 Penekanan pada puss baton dilakukan oleh user

4.5 Program Pada Code Vision AVR

This program was produced by the

CodeWizardAVR V2.05.3 Standard

Automatic Program Generator

© Copyright 1998-2011 Pavel Haiduc, HP InfoTech s.r.l.

http://www.hpinfotech.com

Project :

Version :

Date : 6/28/2012

Author : No Name

Company : No Name

Comments:

```
Chip type           : ATmega32
Program type        : Application
AVR Core Clock frequency: 8.000000 MHz
Memory model        : Small
External RAM size   : 0
Data Stack size     : 512
*****************************************************/

#include <mega32.h>
#include <alcd.h>
#include <spi.h>
#include <ff.h>
#include <sdcard.h>
#include <delay.h>
#include <stdio.h>
#include <string.h>

#define T1_OVF_FREQ 100
#define T1_PRESC 1024L
#define T1_INIT (0x10000L-(_MCU_CLOCK_FREQUENCY_/(T1_PRESC*T1_OVF_FREQ)))

#define ss PORTB.4
unsigned char aa;
void remove();
```

```
void mount_on();
void mk_dir();
void mk_file();
void write_data();
void mount_off();
void take_data(unsigned int len_data);
void reading();



unsigned char data;
FATFS fs;
FIL ftest;
FRESULT res;
char filename[15]="path.txt";
char FBuffer[100]="Fisika UB\n\r";
//char *file;;
//unsigned char data;
char foldername[10]="path";
FRESULT report;
char FBuffer[100];
unsigned int bw;

// Declare your global variables here
interrupt [TIM2_COMP] void timer2_comp_isr(void)
```

```
{
      disk_timerproc();
}

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
// Func7=Out Func6=In Func5=Out Func4=Out Func3=In Func2=In Func1=In Func0=In
// State7=0 State6=T State5=0 State4=0 State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0xB0;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
```

```
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=0xFF
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer1 Stopped
// Mode: Normal top=0xFFFF
// OC1A output: Discon.
// OC1B output: Discon.
```

```
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
//TCCR1B=0x00;
TCCR1B=(1<<CS12)|(1<<CS10);
//TCNT1H=0x00;
TCNT1H=T1_INIT>>8;
//TCNT1L=0x00;
TCNT1L=T1_INIT&0xFF;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
```

```
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
//TIMSK=0x00;
TIMSK=1<<TOIE1;

// USART initialization
// USART disabled
UCSRB=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
```

```
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC disabled
ADCSRA=0x00;

// SPI initialization
// SPI Type: Master
// SPI Clock Rate: 125.000 kHz
// SPI Clock Phase: Cycle Start
// SPI Clock Polarity: Low
// SPI Data Order: MSB First
SPCR=0x52;
SPSR=0x00;

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

// Alphanumeric LCD initialization
// Connections are specified in the
// Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:
// RS - PORTA Bit 0
```

```
// RD - PORTA Bit 1
// EN - PORTA Bit 2
// D4 - PORTA Bit 4
// D5 - PORTA Bit 5
// D6 - PORTA Bit 6
// D7 - PORTA Bit 7
// Characters/line: 16
lcd_init(16);
lcd_clear();
lcd_puts("bisa");
delay_ms(1000);
lcd_clear();

while (1)
      {
      // Place your code here
    ss=0;
mount_on();
mk_dir();
mk_file();
write_data();
mount_off();
//aa = f_read(&ftest, FBuffer, 1, &bw);
mount_on();
```

```
//take_data(10);
//remove();
mount_off();
reading();

//lcd_putchar(aa);
ss=1;
//while(1);
    }
}

void remove()
{
    report=0;
    do
    {
        report=f_unlink(filename);

    }
    while(report!=FR_OK);
}

void mount_on()
```

```
{
        f_mount(0,&fs);
}

void mk_dir()
{
        f_mkdir(foldername);
}

void mk_file()
{
        report=0;
        do
        {
                report=f_open(&ftest,filename, FA_CREATE_ALWAYS | FA_WRITE);


        }
                while(report!=FR_OK);
}

void write_data()
{
        report=0;
```

```
        do
        {
            report=f_write(&ftest, FBuffer, strlen(FBuffer), &bw);
        }
            while(report!=FR_OK);
}

void mount_off()
{
        f_close(&ftest);
        f_mount(0, NULL);
}

void take_data(unsigned int len_data)
{
      unsigned int len_data2;

         //FBuffer[0]=0;
         //for (len_data2=0;len_data2<len_data;len_data2++)
         //while (FBuffer[0]!='E' )
         //while(1)
         //{

              aa= f_read(&ftest, FBuffer, strlen(FBuffer), &bw);
```

```
            aa='a';
             //if (FBuffer[0]=='E') break;

           //   for (data=0;data<10;data++)
           //   {
                    //delay_ms(100);
                  //  lcd_putchar(FBuffer[data]);
                     lcd_putchar(aa);

           //    }
           //   putchar(0x0d);

          //}

}

void reading()
{
     f_mount(0,&fs);
     report=f_open(&ftest,filename, FA_OPEN_EXISTING | FA_READ);
     if (report==FR_NO_PATH)
     {
          lcd_puts("NO PATH");
```

```
}
else if  (report==FR_NO_FILE)
{
      lcd_puts("NO FILE");

}
else
{
      take_data(200);
}
f_close(&ftest);
f_mount(0, NULL);
```

4.6 Cara Kerja

Cara kerja dari hasil kerja praktek ini adalah jika tombol puss baton yang di tekan, maka MMC akan membaca data yang telah tersimpan didalam MMC, dan proses awalnya adalah user menulis data terlebih dahulu lalu data tersebut disimpan dalam bentuk file txt. Kemudian file tersebut akan dirposes oleh MMC dan hasil dari pemrosesan MMC akan ditampilkan pada layar LCD, dan sistem pembacaannya akan di baca tiap suku kata atau per kalimat.

Jika tombol puss baton di tekan lagi maka alat kami akan mereset semua data yang telah tersimpan.