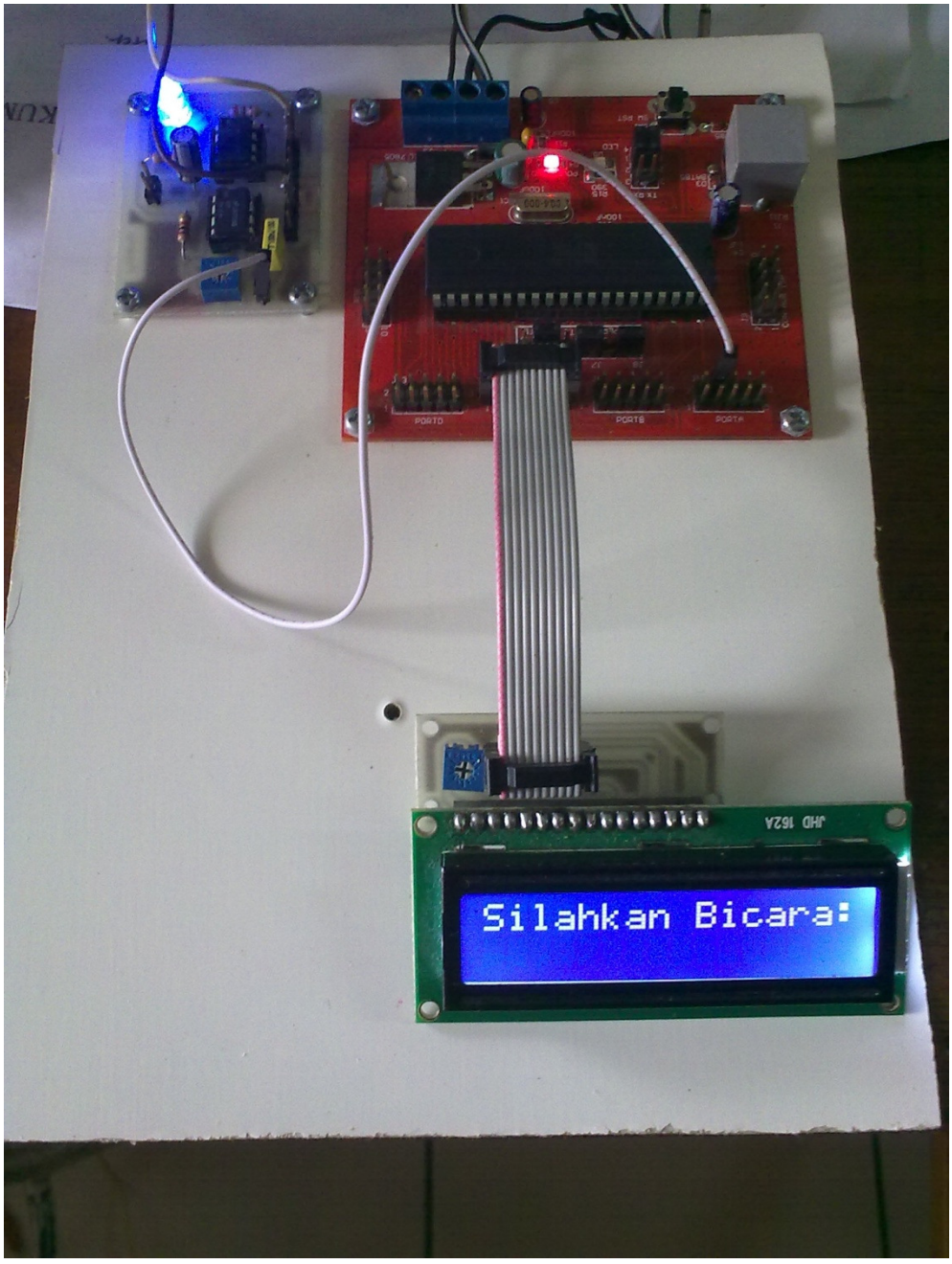
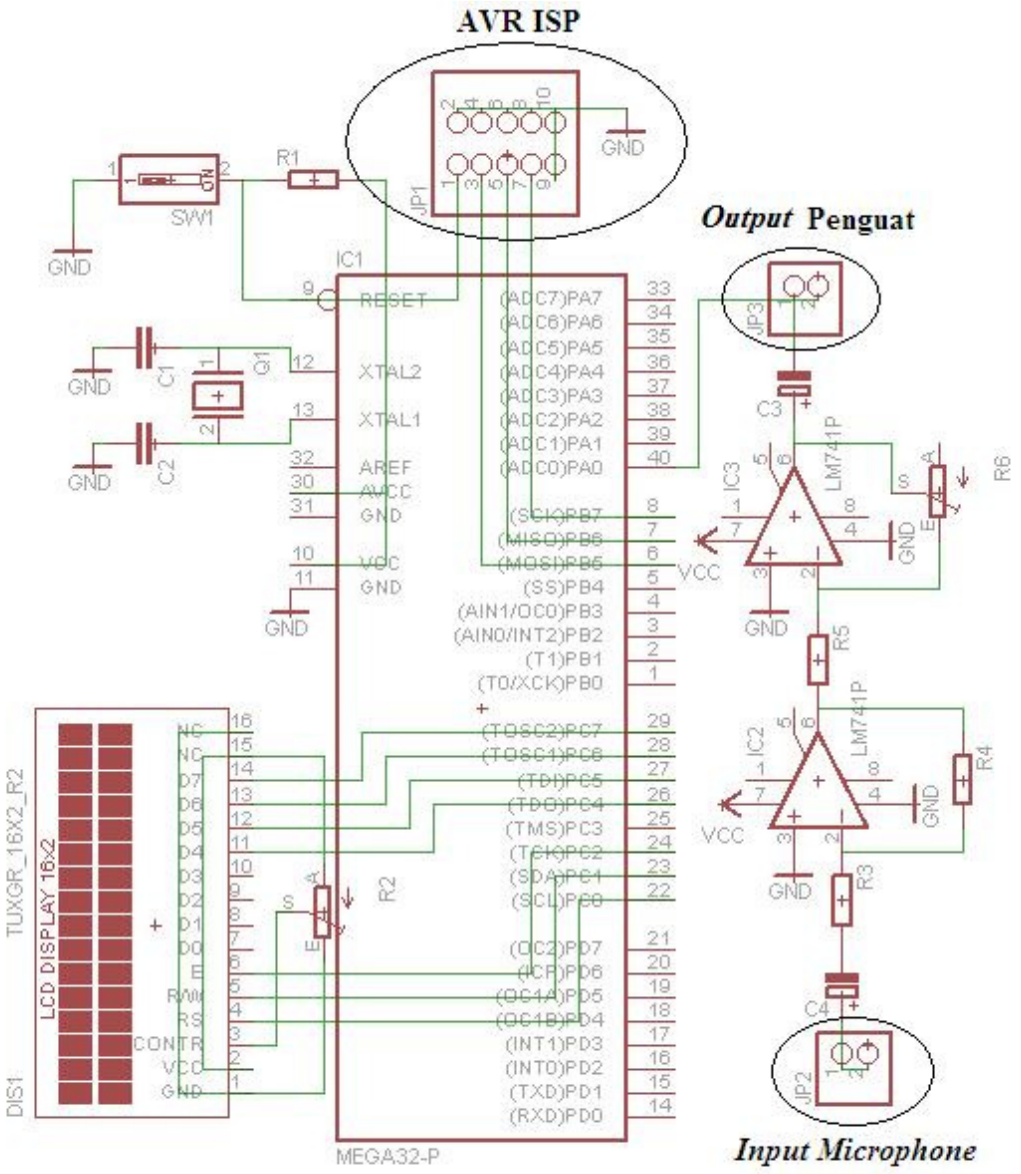
LAMPIRAN A
FOTO ALAT PENGENALAN UCAPAN


Silahkan Bicara:

LAMPIRAN B

SKEMATIK ALAT PENGENALAN UCAPAN


AVR ISP
JP1
GND
SW1
R1
IC1
RESET
XTAL2
XTAL1
AREF
AVCC
GND
VCC
C1
C2
Q1
(ADC7)PA7
(ADC6)PA6
(ADC5)PA5
(ADC4)PA4
(ADC3)PA3
(ADC2)PA2
(ADC1)PA1
(ADC0)PA0
(SCK)PB7
(MISO)PB6
(MOSI)PB5
(SS)PB4
(AIN1/OC0)PB3
(AIN0/INT2)PB2
(T1)PB1
(T0/XCK)PB0
(TOSC2)PC7
(TOSC1)PC6
(TDI)PC5
(TDO)PC4
(TMS)PC3
(TCK)PC2
(SDA)PC1
(SCL)PC0
(OC2)PD7
(ICP)PD6
(OC1A)PD5
(OC1B)PD4
(INT1)PD3
(INT0)PD2
(TXD)PD1
(RXD)PD0
MEGA32-P
Output Penguat
JP3
C3
IC3
LM741P
R6
R5
IC2
R4
R3
C4
JP2
Input Microphone
TUXGR_16X2_R2
LCD DISPLAY 16x2
DIS1
CONTR
R2

LAMPIRAN C
PROGRAM PADA PENGONTROL MIKRO ATMEGA32

PROGRAM UTAMA

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.25.3 Professional
Automatic Program Generator
© Copyright 1998-2007 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 8/15/2010
Author  : F4CG
Company : F4CG
Comments:


Chip type           : ATmega32
Program type        : Application
Clock frequency     : 4.000000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 512
*****************************************************/

#include <mega32.h>
#include <math.h>
#include <stdio.h>
#include <delay.h>

// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>

// Standard Input/Output functions
#include <stdio.h>

#define ADC_VREF_TYPE 0x60

// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}
```

```
// Declare your global variables here
unsigned int k,n,adc[25],j,s,x;
float r[25],i[25],m[25],hasil,hasil2,hasil3,hasil4,hasil5,h1,h2,h3,h4,h5;
eeprom float kanan[25]={3192.7, 106.08, 72.18, 81.76, 50.98, 76.93, 88.87, 56.03, 60.68, 46.23, 49.2, 48.1,
54.48, 54.48, 48.1, 49.2, 46.23, 60.68, 56.03, 88.87, 76.93, 50.98, 81.76, 72.18, 106.08};
eeprom float kiri[25]={4439.3, 229.27, 220.49, 278.26, 193.43, 192.52, 159.28, 217.87, 202.32, 133, 169.44,
170.84, 138.14, 138.14, 170.84, 169.44, 133, 203.32, 217.87, 159.28, 192.46, 193.43, 278.26, 220.49, 229.27};
eeprom float maju[25]={3793.5, 170.19, 126.59, 151.54, 207.31, 108.64, 138.11, 154.18, 130.93, 123.19, 155.44,
118.19, 136.08, 136.08, 118.19, 155.44, 124.19, 130.84, 154.18, 168.11, 108.64, 361.31, 151.54, 126.59, 170.19};
eeprom float mundur[25]={4410.4, 394.04, 268.62, 177.38, 131.38, 105.58, 132.27, 203.04, 141.03, 114.90,
202.45, 147.31, 194.64, 194.64, 147.25, 202.39, 114.9, 141.03, 203.04, 132.27, 105.58, 131.44, 177.38, 268.62,
394.04};
eeprom float stop[25]={5072.8, 151.38, 187, 184.37, 145.77, 146.19, 127.72, 207.98, 85.35, 127.39, 198.06,
119.13, 118.52, 118.52, 119.13, 198.06, 127.39, 85.35, 207.92, 127.72, 146.19, 145.77, 184.37, 187, 151.38};

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=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
PORTB=0x00;
DDRB=0x00;

// 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=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
```

```
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// 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;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: Off
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 9600
UCSRA=0x00;
UCSRB=0x08;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x19;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;
```

```
// ADC initialization
// ADC Clock frequency: 31.250 kHz
// ADC Voltage Reference: AVCC pin
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x87;

// LCD module initialization
riski:
lcd_init(16);
lcd_gotoxy(0,0);
lcd_putsf("Silakan Bicara:");

while (1)
     {
      for (n=0; n<25; n++)
      {       ulang:
              adc[n]=read_adc(0);

              if (adc[n]<=110)
              {goto ulang;
              }

              else goto lanjut;
              lanjut:
              delay_ms(20);

     };

      for (k=0; k<25; k++)
      {       for (j=0; j<25; j++)
              {       r[k]=r[k]+adc[j]*cos(6.28*k*j/25);
                      i[k]=i[k]+adc[j]*sin(6.28*k*j/25);
                      m[k]=sqrt((r[k]*r[k])+(i[k]*i[k]));
              };
      };

      for (s=0;s<25;s++)
      {       hasil=hasil+((m[s]-kanan[s])*(m[s]-kanan[s]));
              h1=sqrt(hasil);

              hasil2=hasil2+((m[s]-maju[s])*(m[s]-maju[s]));
              h2=sqrt(hasil2);

              hasil3=hasil3+((m[s]-kiri[s])*(m[s]-kiri[s]));
              h3=sqrt(hasil3);

              hasil4=hasil4+((m[s]-mundur[s])*(m[s]-mundur[s]));
              h4=sqrt(hasil4);

              hasil5=hasil5+((m[s]-stop[s])*(m[s]-stop[s]));
              h5=sqrt(hasil5);
      };

      if ((h1<h2)&&(h1<h3)&&(h1<h4)&&(h1<h5)&&(h1<375))
```

```
 {lcd_gotoxy(0,1);
  lcd_putsf("kanan");
  delay_ms(2000);
  lcd_gotoxy(0,1);
  lcd_putsf("Accepted");
  delay_ms(2000);
  }

else if ((h2<h1)&&(h2<h3)&&(h2<h4)&&(h2<h5)&&(h2<375))
 {lcd_gotoxy(0,1);
  lcd_putsf("maju");
  delay_ms(2000);
  lcd_gotoxy(0,1);
  lcd_putsf("Accepted");
  delay_ms(2000);
  }

else if ((h3<h1)&&(h3<h2)&&(h3<h4)&&(h3<h5)&&(h3<375))
 {lcd_gotoxy(0,1);
  lcd_putsf("kiri");
  delay_ms(2000);
  lcd_gotoxy(0,1);
  lcd_putsf("Accepted");
  delay_ms(2000);
  }

else if ((h4<h1)&&(h4<h2)&&(h4<h3)&&(h4<h5)&&(h4<375))
 {lcd_gotoxy(0,1);
  lcd_putsf("mundur");
  delay_ms(2000);
  lcd_gotoxy(0,1);
  lcd_putsf("Accepted");
  delay_ms(2000);
  }

else if ((h5<h1)&&(h5<h2)&&(h5<h4)&&(h5<h3)&&(h5<375))
 {lcd_gotoxy(0,1);
  lcd_putsf("stop");
  delay_ms(2000);
  lcd_gotoxy(0,1);
  lcd_putsf("Accepted");
  delay_ms(2000);
  }

 else
 {lcd_gotoxy(0,1);
  lcd_putsf("denied");
  delay_ms(1000);
  lcd_clear();
   };

 delay_ms(2000);
 for (x=0; x<25; x++)
 { lcd_clear();
   adc[x]=0;
   r[x]=0;
   i[x]=0;
```

```
        m[x]=0;
        hasil=0;
        hasil2=0;
        hasil3=0;
        hasil4=0;
        hasil5=0;
       };
       goto riski;

      // Place your code here

      };
}
```

PROGRAM PEMBUATAN DATABASE

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.25.3 Professional
Automatic Program Generator
© Copyright 1998-2007 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 8/10/2010
Author  : F4CG
Company : F4CG
Comments:


Chip type           : ATmega32
Program type        : Application
Clock frequency     : 4.000000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 512
*****************************************************/

#include <mega32.h>
#include <math.h>
#include <stdio.h>
#include <delay.h>

// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>

// Standard Input/Output functions
#include <stdio.h>

#define ADC_VREF_TYPE 0x60

// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}
```

```
// Declare your global variables here
unsigned int k,n,adc[25],j;
float r[25],i[25],m[25];
char buffer[33],buffer2[33],buffer3[33],buffer4[33],buffer5[33];

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=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
PORTB=0x00;
DDRB=0x00;

// 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=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
```

```
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// 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;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: Off
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 9600
UCSRA=0x00;
UCSRB=0x08;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x19;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC Clock frequency: 31.250 kHz
// ADC Voltage Reference: AVCC pin
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x87;

// LCD module initialization
lcd_init(16);
```

```
while (1)
    {
    for (n=0; n<25; n++)
    {       ulang:
            adc[n]=read_adc(0);


            if (adc[n]<=110)
            {goto ulang;
            }

            else goto lanjut;
            lanjut:
            delay_ms(20);
    };

    for (k=0; k<25; k++)
    {       for (j=0; j<25; j++)
            {       r[k]=r[k]+adc[j]*cos(6.28*k*j/25);
                    i[k]=i[k]+adc[j]*sin(6.28*k*j/25);
                    m[k]=sqrt((r[k]*r[k])+(i[k]*i[k]));

                   sprintf(buffer,"%3.1f %3.1f %3.1f %3.1f %3.1f",m[0],m[1],m[2],m[3],m[4]);
                    lcd_gotoxy(0,0);
                    lcd_puts(buffer);
                    delay_ms(500);

                    sprintf(buffer2,"%3.1f %3.1f %3.1f %3.1f %3.1f ",m[5],m[6],m[7],m[8],m[9]);
                    lcd_gotoxy(0,0);
                    lcd_puts(buffer2);
                    delay_ms(500);

                    sprintf(buffer3,"%3.1f %3.1f %3.1f %3.1f %3.1f ",m[10],m[11],m[12],m[13],m[14]);
                    lcd_gotoxy(0,0);
                    lcd_puts(buffer3);
                    delay_ms(500);

                   sprintf(buffer4,"%3.1f %3.1f %3.1f %3.1f %3.1f ",m[15],m[16],m[17],m[18],m[19]);
                    lcd_gotoxy(0,0);
                    lcd_puts(buffer4);
                    delay_ms(500);

                   sprintf(buffer5,"%3.1f %3.1f %3.1f %3.1f %3.1f ",m[20],m[21],m[22],m[23],m[24]);
                    lcd_gotoxy(0,0);
                    lcd_puts(buffer5);
                    delay_ms(500);
            };
    };

        // Place your code here

    };
}
```