BAB V

PENUTUP

A. Kesimpulan

Berdasarkan penelitian yang telah dilakukan dalam proses pembuatan PROTOTYPE RANCANG BANGUN SISTEM MONITORING DETAK JANTUNG MENGGUNAKAN JARI BERBASIS MIKROKONTROLER ATMEGA328 dapat disimpulkan bahwa :

1. Alat ini di bangun dengan menggunakan pemograman arduino dan juga menggunakan server agar masukan dan pengeluaran data dapat di ketahui oleh user, terdapat 4 menu dan program yang ada pada alat diantaranya :
    a) Server, dalam bagian server user dapat melihat input dan output data detak jantung
    b) Sensor dapat mendeteksi detak jantung melalui jari.
    c) Mikrokontroler ATMEGA328 dapat diaktif kan dengan catu daya eksternal.
    d) Proses ini menampil kan data detak jantung berupa angka 100/110 dengan keadaan normal atau keadaan setelah beraktifitas

2. Yang dilakukan pada alat ini sudah dapat berjalan dengan baik, sesuai dengan fungsinya yaitu mengeluar kan data detak jantung menggunakan jari berupa angka baets per menit.
3. Berdasarkan hasil pengujian dari kuisioner di dapatkan Berdasarkan hasil dari tabel diatas, dapat diperoleh persentase penilaian terhadap sistem yaitu : B = 84/130*100 = 69%, C = 36/130*150 = 30%, K = 1/130*100 = 1%.

B. Saran

Saran yang dapat penulis berikan untuk penelitian lebih lanjut agar aplikasi ini menjadi lebih sempurna antara lain :

1. Penggunaan probe sangat di pengaruhi oleh gerakan tubuh (jari tanggan) sehingga perlu pembuatan probe yang lebih fleksibel dan kokoh.
2. Desain perlu ditingkatkan (lebih compack) agar mudah pemakain
3. Membuat alat pendeteksi detak jantung dengan menggunakan metode plethysmogph yang dapat sekaligus mengetahui kadar oksigen dalam darah.

DAFTAR PUSTAKA

Alan Trevennor. 2013. *Practical AVR Microcontrollers. Apress. United States of America*

Andreyanto 2012. *Menjelas kan tentang flowcart http://andreyanto-gunadarma.blogspot.co.id*

Aristyaningtyas, Detak Jantung. Diambil pada tanggal 07 May 2015, dari *www.ratih52.web.unej.ac.id*

Dhananjay V. Garde. 2001. *Programming And Customizing The AVR Microcontroller. McGraw-Hill. United States of America*

Erliyanto, dkk 2008. *Menjelaskan Bahwa Memonitoring Jantung Sangat Penting Dengan Penampil LCD*

Faisal 2008. *Tentang Rancang Bangun Sistem Pengukuran Denyut Jantung Dengan Menggunakan Sensor Fotodioda dan Penamil LCD Berbasis Mikrokontroler ATmega98S52*

Heruryanto, dkk 2014. *Merancang dan Membuat Sistem Pengukuran Denyut Jantung Berbasis Mikrokontroler ATmega8535*

Julien Byle. 2013. *C Programming for Arduino. Packt Publishing Ltd. Birmingham UK*

Suhanta,2004. *Mikrokontroler atmel. Diambil Pada 13 Oktober 2014, dari www.atmel.com*

Ratih 2010. *Tentang Sistem Pengukuran Denyut Jantung Berbasis mikrokontroler ATmega8535*

Rusmin, (2012).*Getting Start Pulse Sensor from : URL : HYPERLINK http://pulsesensor.myshopify.com/blogs/news*

Winoto, ATmega328. Diambil pada tahun 2008, dari *www.katalogbiobses.com*

LAMPIRAN

```
#include <Wire.h>  // Comes with Arduino IDE

#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3,
POSITIVE);  // Set the LCD I2C address


//  VARIABLES

int pulsePin = A0;                  // Pulse Sensor purple
wire connected to analog pin 0

int blinkPin = 13;                  // pin to blink led at
each beat

int fadePin = 5;                  // pin to do fancy classy
fading blink at each beat

int fadeRate = 0;                  // used to fade LED on
with PWM on fadePin



// these variables are volatile because they are used during
the interrupt service routine!

volatile int BPM;                  // used to hold the
pulse rate

volatile int Signal;                // holds the incoming
raw data

volatile int IBI = 600;               // holds the time
between beats, the Inter-Beat Interval

volatile boolean Pulse = false;     // true when pulse wave
is high, false when it's low
```

```
volatile boolean QS = false;          // becomes true when
Arduoino finds a beat.


void setup(){
  pinMode(blinkPin,OUTPUT);           // pin that will blink
to your heartbeat!
  pinMode(fadePin,OUTPUT);           // pin that will fade to
your heartbeat!
  lcd.begin(16,2);
  for(int i = 0; i< 3; i++)
  {
    lcd.backlight();
    delay(250);
    lcd.noBacklight();
    delay(250);
  }
  lcd.backlight(); // finish with backlight on

  Serial.begin(115200);                // we agree to talk
fast!
  interruptSetup();                 // sets up to read Pulse
Sensor signal every 2mS
   // UN-COMMENT THE NEXT LINE IF YOU ARE POWERING The Pulse
Sensor AT LOW VOLTAGE,
   // AND APPLY THAT VOLTAGE TO THE A-REF PIN

   //analogReference(EXTERNAL);

   lcd.setCursor(0,0);

  lcd.print("Loading..");

  delay(1000);

  lcd.setCursor(0,1);

  lcd.print("System..");

  delay(3000);

}

void loop(){

  lcd.clear();

  lcd.setCursor(0,0);

  lcd.print(BPM);

  lcd.setCursor(0,1);

  lcd.print("Beat/Minute");


  Serial.println(BPM);

  delay(50);


  sendDataToProcessing('S', Signal);      // send Processing
the raw Pulse Sensor data

  if (QS == true){                        // Quantified Self
flag is true when arduino finds a heartbeat
```

```
        fadeRate = 255;                    // Set 'fadeRate'
Variable to 255 to fade LED with pulse
        sendDataToProcessing('B',BPM);   // send heart rate
with a 'B' prefix
        sendDataToProcessing('Q',IBI);     //  send  time
between beats with a 'Q' prefix
        QS = false;                          // reset the
Quantified Self flag for next time
     }


  ledFadeToBeat();


  delay(150);                            //  take a break


}
void ledFadeToBeat(){
    fadeRate -= 15;                        //  set LED fade
value
    fadeRate = constrain(fadeRate,0,255);   //   keep LED
fade value from going into negative numbers!
    analogWrite(fadePin,fadeRate);          //  fade LED
  }
void sendDataToProcessing(char symbol, int data ){
    Serial.print(symbol);                  // symbol prefix
tells Processing what type of data is coming
```

```
    Serial.println(data);               // the data to send
culminating in a carriage return

  }

volatile int rate[10];                      // used to hold
last ten IBI values

volatile unsigned long sampleCounter = 0;           // used
to determine pulse timing

volatile unsigned long lastBeatTime = 0;            // used
to find the inter beat interval

volatile int P =512;                        // used to find
peak in pulse wave

volatile int T = 512;                       // used to find
trough in pulse wave

volatile int thresh = 512;                  // used to find
instant moment of heart beat

volatile int amp = 100;                     // used to hold
amplitude of pulse waveform

volatile boolean firstBeat = true;          // used to seed
rate array so we startup with reasonable BPM

volatile boolean secondBeat = true;         // used to seed
rate array so we startup with reasonable BPM

void interruptSetup(){

  // Initializes Timer2 to throw an interrupt every 2mS.

  TCCR2A = 0x02;      // DISABLE PWM ON DIGITAL PINS 3 AND
11, AND GO INTO CTC MODE

  TCCR2B = 0x06;     // DON'T FORCE COMPARE, 256 PRESCALER
  OCR2A = 0X7C;      // SET THE TOP OF THE COUNT TO 124 FOR
500Hz SAMPLE RATE

  TIMSK2 = 0x02;      // ENABLE INTERRUPT ON MATCH BETWEEN
TIMER2 AND OCR2A

  sei();               // MAKE SURE GLOBAL INTERRUPTS ARE
ENABLED

}

// THIS IS THE TIMER 2 INTERRUPT SERVICE ROUTINE.

// Timer 2 makes sure that we take a reading every 2
miliseconds

ISR(TIMER2_COMPA_vect){                        // triggered
when Timer2 counts to 124

    cli();                                      // disable
interrupts while we do this

    Signal = analogRead(pulsePin);              // read the
Pulse Sensor

    sampleCounter += 2;                            // keep
track of the time in mS with this variable

    int N = sampleCounter - lastBeatTime;       // monitor
the time since the last beat to avoid noise

//  find the peak and trough of the pulse wave

    if(Signal < thresh && N > (IBI/5)*3){        // avoid
dichrotic noise by waiting 3/5 of last IBI

        if (Signal < T){                        // T is the
trough

            T = Signal;                            // keep
track of lowest point in pulse wave
```

```
          }
        }

    if(Signal > thresh && Signal > P){            // thresh
condition helps avoid noise
        P = Signal;                             // P is the
peak
       }                                           // keep
track of highest point in pulse wave

  //  NOW IT'S TIME TO LOOK FOR THE HEART BEAT
  // signal surges up in value every time there is a pulse
if (N > 250){                                      // avoid
high frequency noise
  if  (  (Signal  >  thresh)  &&  (Pulse  ==  false)  &&  (N  >
(IBI/5)*3) ){
    Pulse = true;                               // set the
Pulse flag when we think there is a pulse
    digitalWrite(blinkPin,HIGH);                // turn on
pin 13 LED
    IBI = sampleCounter - lastBeatTime;         // measure
time between beats in mS
    lastBeatTime = sampleCounter;                  // keep
track of time for next pulse
```

```
         if(firstBeat){                           // if it's
the first time we found a beat, if firstBeat == TRUE
             firstBeat = false;                     // clear
firstBeat flag
             return;                            // IBI value
is unreliable so discard it
            }
         if(secondBeat){                          // if this
is the second beat, if secondBeat == TRUE
            secondBeat = false;                     // clear
secondBeat flag
               for(int i=0; i<=9; i++){          // seed the
running total to get a realisitic BPM at startup
                    rate[i] = IBI;
                    }
            }

    // keep a running total of the last 10 IBI values
    word runningTotal = 0;                      // clear the
runningTotal variable

    for(int i=0; i<=8; i++){                // shift data in
the rate array
          rate[i] = rate[i+1];               // and drop the
oldest IBI value
```

```
          runningTotal += rate[i];          // add up the 9
oldest IBI values
        }


    rate[9] = IBI;                          // add the
latest IBI to the rate array
    runningTotal += rate[9];                // add the
latest IBI to runningTotal
    runningTotal /= 10;                     // average the
last 10 IBI values
    BPM = 60000/runningTotal;               // how many
beats can fit into a minute? that's BPM!
    QS = true;                              // set
Quantified Self flag
    // QS FLAG IS NOT CLEARED INSIDE THIS ISR
    }
}
  if (Signal < thresh && Pulse == true){       // when the
values are going down, the beat is over
      digitalWrite(blinkPin,LOW);           // turn off pin
13 LED
      Pulse = false;                        // reset the
Pulse flag so we can do it again
      amp = P - T;                          // get
amplitude of the pulse wave
```

```
      thresh = amp/2 + T;                      // set thresh
at 50% of the amplitude
      P = thresh;                             // reset these
for next time
      T = thresh;
     }
  if (N > 2500){                                  // if 2.5
seconds go by without a beat
      thresh = 512;                            // set thresh
default
      P = 512;                                      // set P
default
      T = 512;                                      // set T
default
      lastBeatTime = sampleCounter;            // bring the
lastBeatTime up to date
      firstBeat = true;                      // set these to
avoid noise
      secondBeat = true;                      // when we get
the heartbeat back
     }
  sei();                                          // enable
interrupts when youre done!
}// end isr
```