5-2016

Heart Beat Monitor

Ivan Mireles
Trinity University, imireles@trinity.edu

Sneha Pottian
Trinity University, spottian@trinity.edu

Pledged,

Project Report

Heart beat monitor

Ivan Mireles & Sneha Pottian

Mechatronics
ENGR 4367
Dr. Nickels
5/1/16

I. Table of Contents

II. Design Summary

This project is used to detect the number of heartbeats per minute from the fingertip of a patient. It is also meant to display how a heart beats based off the signal it picks up from the pulse through an analog counter. Once the user turns on the power button, they will be instructed to place their finger on the sensing tip (ref. To Fig. 1).Once the user has placed their finger on the pulse sensor, the system will require that the start button be pressed to start counting up the heart beats. As the heart beats are sensed the counter will start moving in a circle at increments that match the beat along with beeps from a speaker to visually display it. The sensor will take about 15s to count the heartbeats and once it is done, the clock will stop at a number and visually display it on an LCD screen in BPM (beats per minute). If the heart rate is too high or too low, the speaker goes off warning the user. After a set number of seconds, the system goes back to the default setting waiting for the next input. Figure 1 displays how each individual elements connects to the overall system. From the diagram, one can see that the visual counter is controlled by an Arduino which takes the input from the PIC.

III. System Details

The infrared LED sends signal to the photodiode. The voltage across the photodiode varies based on how much infrared it picks up. This combination of LED and photodiode is set up so that the LED only emits to the top, and the detector only picks up from the top. When a finger is placed on top of both components, the only light the detector sees is through the finger. The blood in the finger absorbs some of the infrared light, and the amount of blood in the finger affects the amount of light picked up by the detector, which, as stated, affects the voltage across the detector. When there is a heartbeat, there is a change in the amount of blood in the finger, which causes a change in voltage. Since heartbeats are periodic, the voltage across the detector under the influence of the finger's heartbeat results in a periodic signal. This signal is fairly small with a maximum amplitude of ~70mV, as seen in Figure 4. It is also irregular in shape, unlike a sine wave, so some conditioning was necessary to "clean" the signal up. Figure 3 in the Appendix contains the signal conditioning circuit which consists of a bandpass filter made of resistors and capacitors and two operational amplifiers that enhances the signal enough to go from -5V to 5V. This adjusted signal is what connects to the PIC, but since the PIC is only powered from 0 to 5V, the signal changes so that it cannot go below 0V.

The toggle-switch power button is what turns on the infrared LED and prompts the LCD to begin the default setting of telling the user to place the finger on the sensors. After the finger is placed, the user may push the normally-open start button to activate the counter sequence. The system waits 3 seconds to let the signal settle into a signal based only on the heartbeat since the finger moving to cover the sensor affects the signal. After the 3 seconds, the counter begins

counting the number of pulses from the input signal for 15 seconds, and multiplies the counter by 4 to convert the number to BPM. As there is a pulse, the PIC sends a digital signal to the Arduino to turn the stepper one step determined by the program. This serves as a visual display of the pulses the finger produces as the PIC is counting these pulses. After the counter is converted, the number is displayed on the LCD. If the heart rate is not between the average rate of 60-100 BPM, the buzzer will sound alerting the user that the heart rate is not within the expected range. It holds this message for 3 seconds before returning to the default setting.

IV. Design Evaluation

According to table 1, we were able to meet all the functional element requirements for the project. The output display that was used in the project was a 16x2 LCD screen that required some research to determine the appropriate connections to the PIC as well as the PICBASIC code that was compatible with the PIC16F88. Unlike the Arduino, it did not need any initialization or pin assignments since it already had some pre-determined connections. The audio output was the piezo speaker that was used to warn the patient when their heartbeat was high or low. The manual user input were the two switches that were used to turn on the system and to start the counting process. The automatic sensor was the infrared sensors. This also required some research on how to test them, an analysis on the minimum distance required to detect the infrared as well as the current variation in them with respect to the amount of infrared being detected.

The stepper motor was the actuator that was used as a visual display of the heartbeat. It incremented its steps for every heartbeat movement. The motor was controlled on the Arduino for every input signal from the PIC. That is, whenever a high was detected, the PIC would send a digital high to the Arduino which would then cause the motor to rotate by four steps. We made it four because the total number of steps on the motor required to complete one revolution was 100 steps. The heart beats are measured over a period of 15s meaning that the total number of heart beats in that range is 25 (considering the maximum is 100 bpm). In order to get 25 steps to complete one revolution, 4 steps have to be taken at once.

In order to control the whole system, a PIC16F88 was used to interface the inputs and the outputs along with the Arduino. Most of the control was done through picBasic on mircocode to control the main inputs and outputs. The Arduino program was used to control the stepper motor that was used as a counter.

Table 1: The functional elements used in the design.

A	Output Display	-LCD
B	Audio output device	-speaker
C	Manual user input	- start button - power button
D	Automatic sensor	-IR photo detector
E	Actuators, mechanisms and hardware	- Stepper motor for the counter
F	Logic, processing and Control	- microcode - Arduino

V. Special Parts

Infrared LED - produces an electromagnetic wave with wavelength between 940nm-950nm. On Mouser Electronics, the device costs $0.70 if bought with individually.

Infrared semiconductor - serves as semiconductor collecting infrared light and increasing voltage as amount of collected light increases. On Mouser Electronics, the device costs $0.48.

VI. Lessons Learned

The way this device was powered could be improved, especially since the stepper motor requires a different voltage than the PIC and the rest of the circuit. A voltage regulator to 5V, another one to -5V, and a single 9V power source would have sufficed for powering the entire device. Another helpful tip is the functionality of the motor.

Another helpful hint when attempting this project is to be aware of the sensitivity of the photo receiver. This system significantly amplifies small signals with low frequencies. For this reason, every little motion by the finger is picked up by the detector, and taken as a pulse by the PIC. This system would be improved if it the sensors were well grounded so as to make the device more user friendly and robust.

The project took much longer than expected due to minor errors such as damaged PICs and LCDs. In these cases, it is quite hard to determine what the problem is until you decide to change the device out. We also ran across some problems with powering the op-amps because of the given specifications. All the above problems had slowed us down on our project. It would have put us in a better place if we started off earlier to leave a lot of room for errors and as well as perfecting the

project. It would have been helpful to also go through the specifications of a device thoroughly before making any assumptions and wiring it up.

VII.References

[1] Blum, Jeremy. "Arduino - Stepper unipolar circuit". *Arduino.cc*. N.p., 2016. Web. 2 May 2016.

VIII.Appendix

Appendix A: Figures

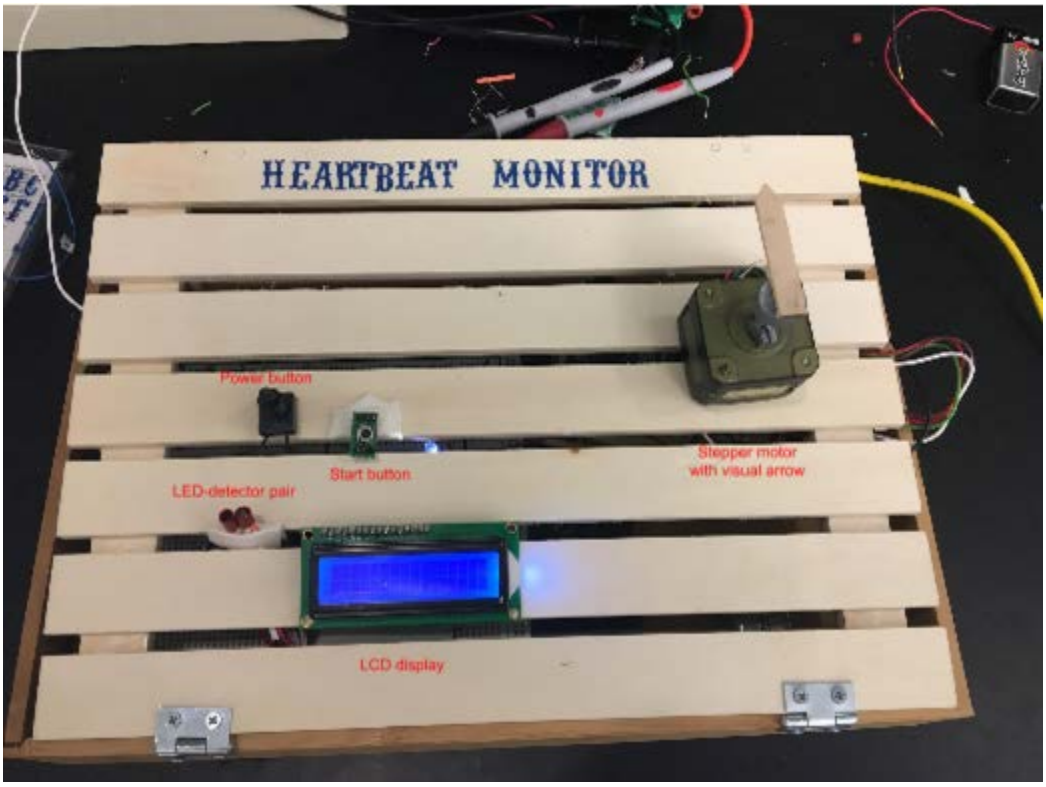

Figure 1: Top view picture of the Heartbeat Monitor system.

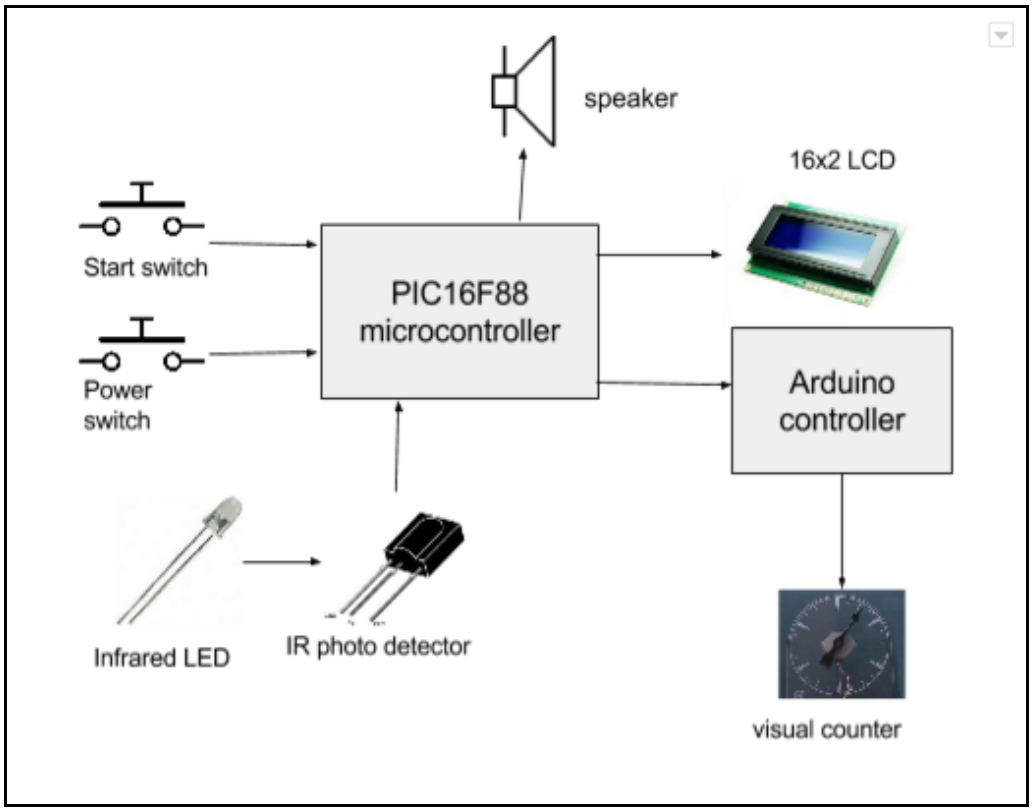

Figure 2: Functional diagram of the heartbeat system

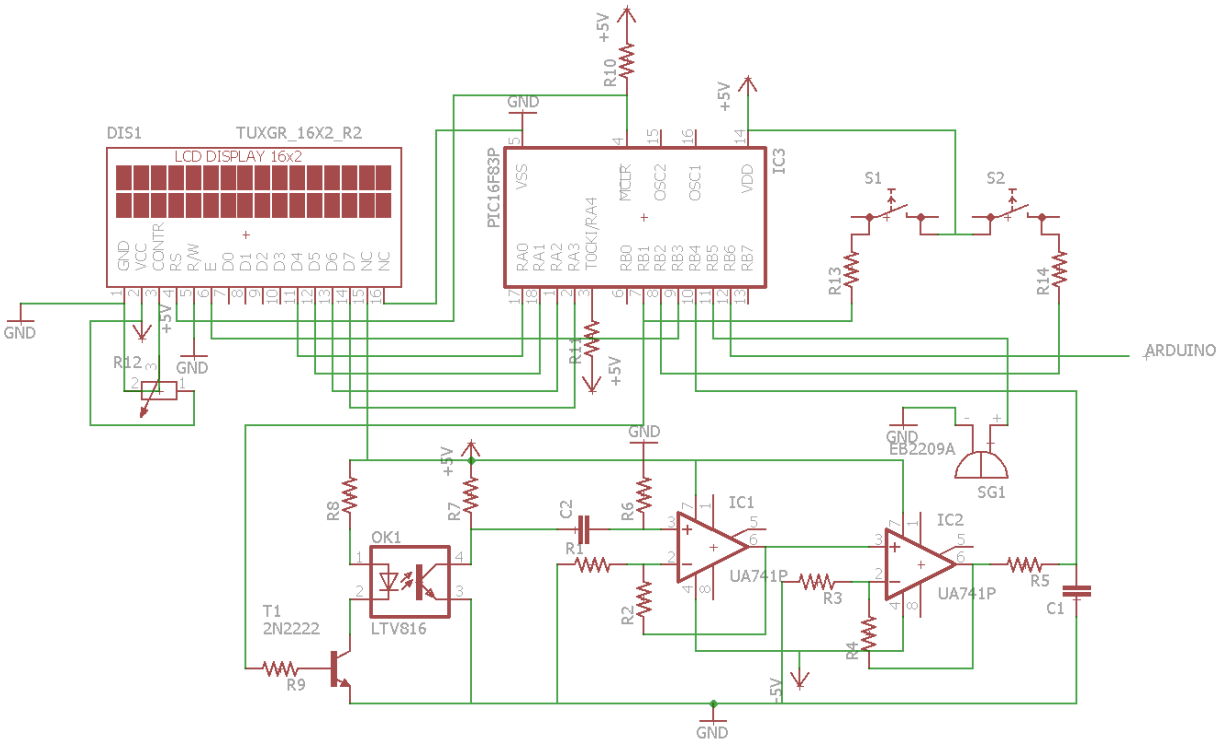

Figure 3: The wiring diagram of the PIC16F88 to it's input and outputs

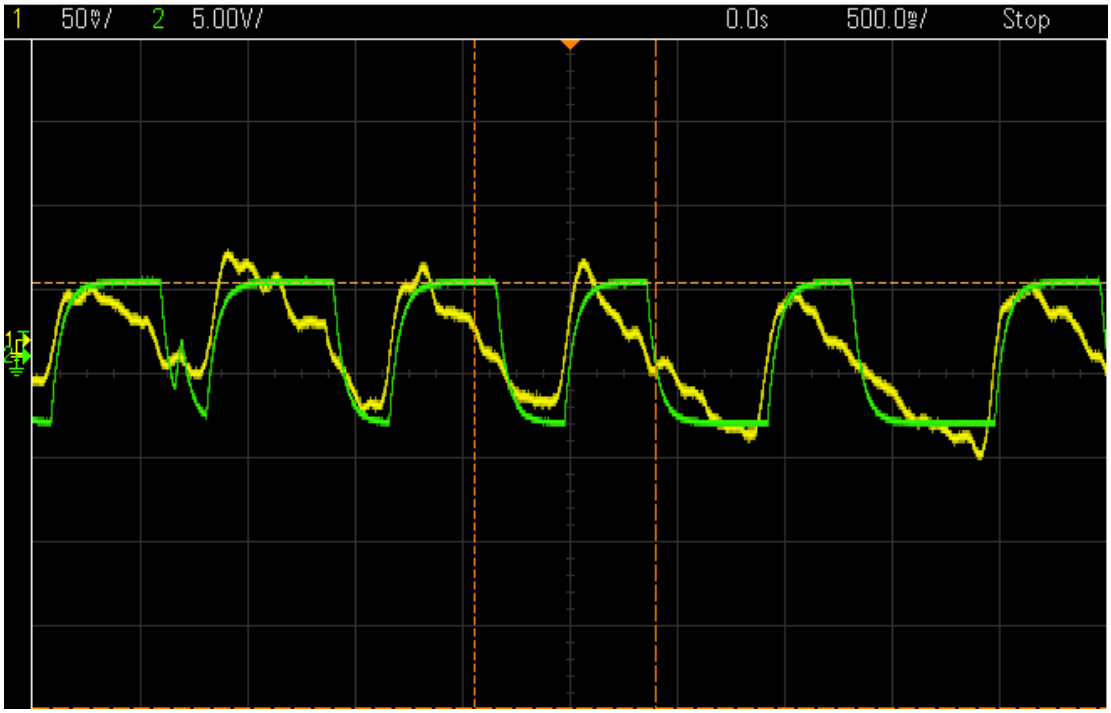

Figure 4: The scope showing the first amplification stage with a gain of 100V/V with the sensor input (yellow) and the amplified signal (green).

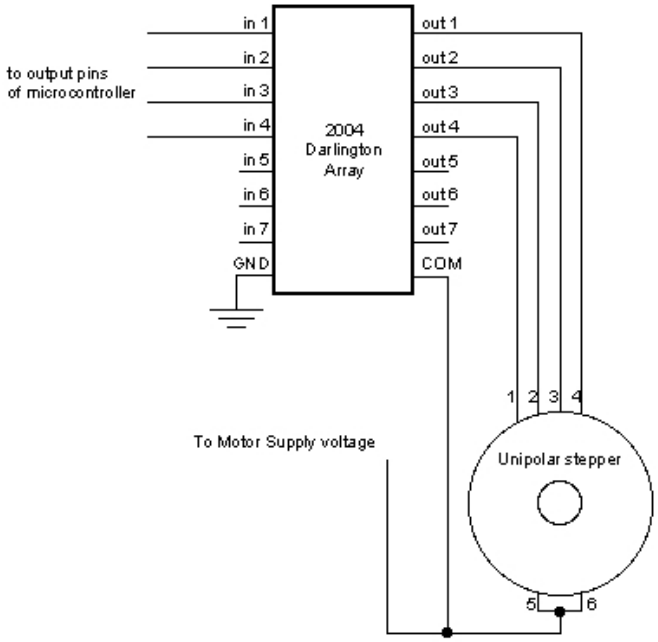

Figure 5: The stepper motor schematic used [1]

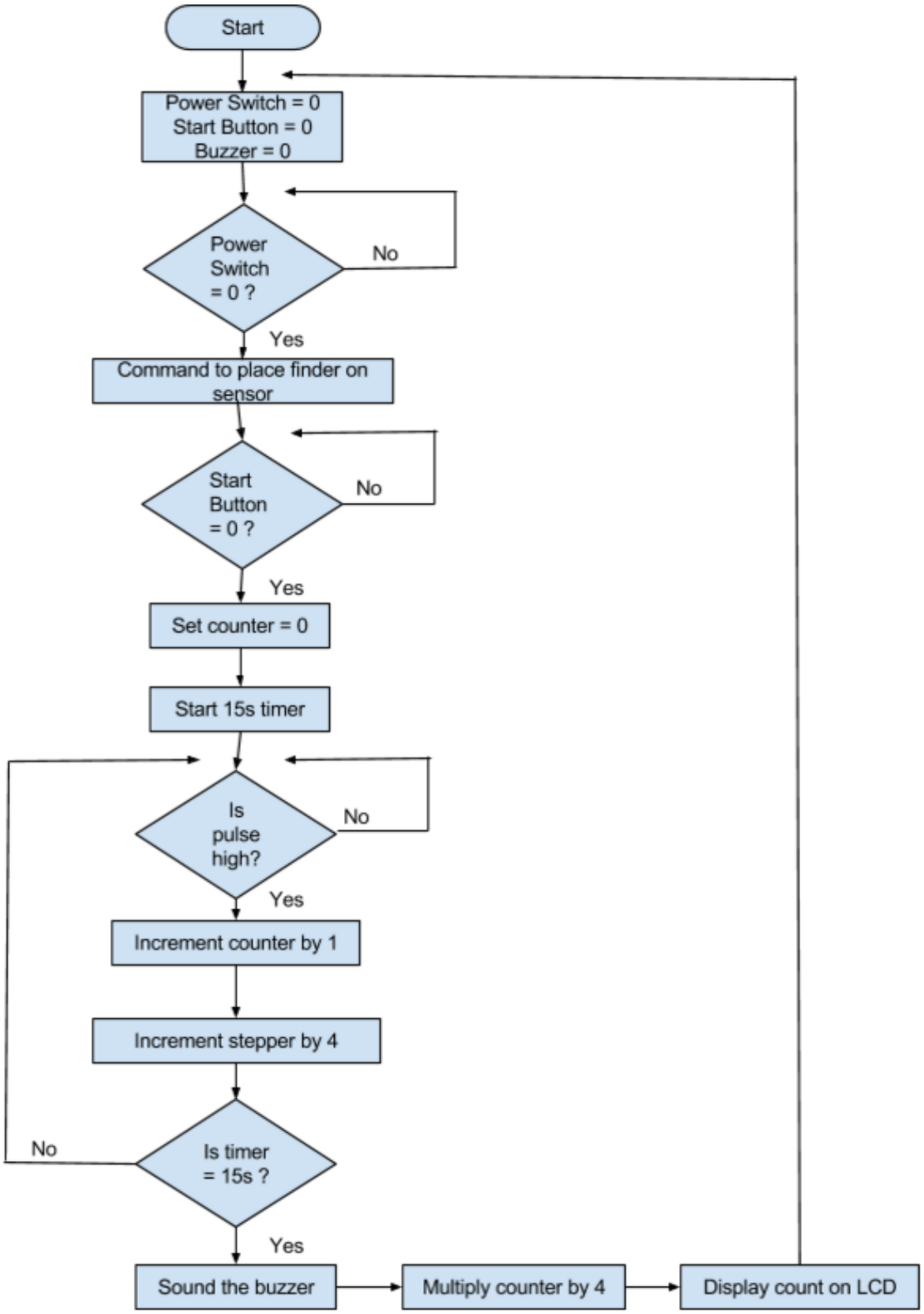

Figure 6: The flowchart diagram of the whole system.

Appendix B: Code

PIC Code:

```
'*****************************************************************
'*  Name    : heartbeat.bas                                      *
'*  Author  : Ivan Mireles, Sneha Pottian                        *
'*  Notice  : Copyright (c) 2016                                 *
'*          : All Rights Reserved                                *
'*  Date    : 4/20/2016                                          *
'*  Version : 1.0                                                *
'*  Notes   :  Program that reads user's heartbeat from finger   *
'*          :  tip and displays in BPM on LCD.                   *
'*****************************************************************


power_b     Var PORTB.2
start_b     var PORTB.1
heartbeat   var PORTB.4
stepper     var PORTB.6
buzzer      var PORTB.5
counter     var BYTE
timer       var byte

' Define configuration settings (different from defaults)
#CONFIG
    __CONFIG _CONFIG1, _INTRC_IO & _PWRTE_ON & _MCLR_OFF & _LVP_OFF
#ENDCONFIG

' Set the internal oscillator frequency to 8 MHz
DEFINE OSC 8
OSCCON.4 = 1
OSCCON.5 = 1
OSCCON.6 = 1

ansel = 0

main:
while (1)
  'initialization
  low stepper
  counter = 0
  timer = 0
  low buzzer
  if (power_b == 1) then
     lcdout , $FE, 1, "Place finger", $FE, $C0, "on sensor" 'Default state
    if (start_b ==1) then
      stepper = heartbeat 'sets stepper to be heartbeat input, controlled by
Arduino
      lcdout $FE, 1,"Adjusting" 'This is to wait for the signal to settle
```

```
      pause 3000
      lcdout $FE, 1,"Checking pulse.."

     'Counter variable counting pulses for 15 sec then storing the value in
BPM
     COUNT heartbeat, 15000, counter
      counter = counter * 4

      'if-then-else ladder determining heartrate status
      if (counter < 60) then
        lcdout $FE, 1,"Low heart rate!", $FE, $C0, dec counter, "BPM"
        high buzzer
        pause 1000
        low buzzer

      elseif (counter > 100) then
        lcdout $FE, 1,"High heart rate!", $FE, $C0, dec counter, "BPM"
        high buzzer
        pause 1000
        low buzzer

      else
      lcdout $FE, 1, "Heart rate:", $FE, $C0, DEC counter, "BPM"
      endif
    pause 3000
    endif
    else
      lcdout $FE, 1
  endif
Wend
```

Arduino code:

```
#include <Stepper.h>

const int stepsPerRevolution = 100;  // change this to fit the number of
steps per revolution
int button = 7;
int state;
int led = 13;

// initialize the stepper library on pins 8 through 11:
Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11);

void setup() {

  myStepper.setSpeed(60); // set the speed at 60 rpm:
  pinMode(button, INPUT);
```

```
  pinMode(led, OUTPUT);

  Serial.begin(9600); // initialize the serial port:

}

void loop() {
  state = digitalRead(button);
  Serial.print("Button state is: ");
  Serial.print(state);
  Serial.print('\n');


 if (state == HIGH)
 {digitalWrite(led, HIGH);
 myStepper.step(4);
 }
 else
 {digitalWrite(led, LOW);
 }
//writing to motor
 delay(600);

}
```