ECE 216 - μprocessors
Final Project Report

Eddie Samuels & Jeremy Warner
University of Rochester

December 16, 2014

Abstract

For our ECE 216 Final Project, we constructed a very low cost heart rate monitor. Using a phototransistor, fluctuations in light absorbed across the user's finger are detected and amplified. These fluctuations of light absorption correspond to variance in the user's blood flow, which mirrors the user's heart beat. This HR signal is passed into a microcontroller which samples the analog signal and records the data digitally. Then, it calculates the total rising edges in the current history buffer, to get the total beats per buffer. Finally, it extrapolates what the users beats per minute (BPM) is, based on the sampling rate of the ADC and the size of the history buffer, and displays this number (BPM) in binary on an array of LEDs for the user to see.

1 The Proposal

Our project will be to create a heart rate monitor, using the PIC microcontroller fundamentals we have learned about thus far in ECE 216. Heart rate monitors can be found in many common devices today, and are also essential to monitoring the health of a user. We believe that the construction of a heart rate monitor not only demonstrates knowledge of the fundamentals taught in ECE 216, but also the ability to apply these concepts to modern applications.

Common applications of heart rate sensors or monitors include patient analysis in hospitals and personal health fitness (e.g Apple Watch or Polar Fitness Meter). A new era where the user is gaining control of the reigns of their personal health information is coming, one where people are empowered to make informed decisions on health care without the help of overworked doctors, lessening the strain on a bloated health care system. Applications such as the heart rate monitor are essential to this new user-geared health enrichment movement.

Figure 1: Example Modern Heart Rate Band

This project will also be useful for our senior design project, where we are designing a wireless network with heart rate signals from numerous patients clustered around a hub. We believe that getting started early in this project will allow for greater success in the future.

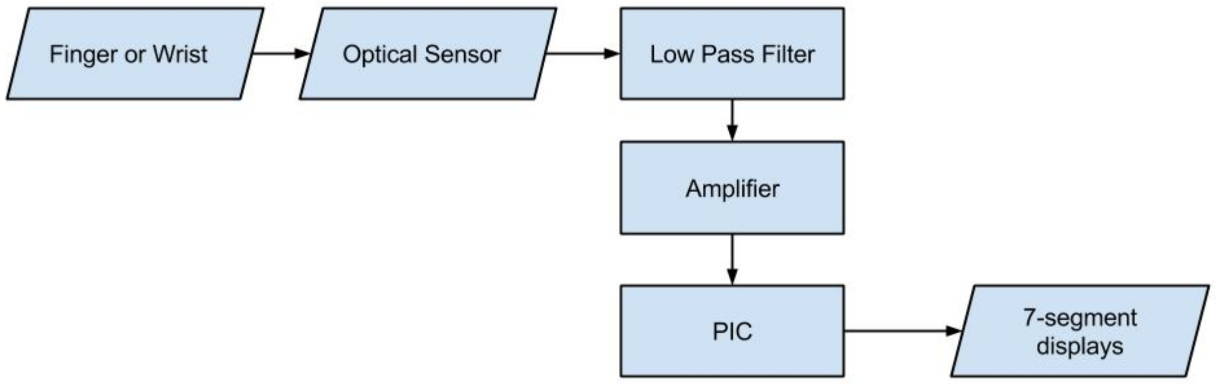
There are several on-line tutorials that teach you how to build a sensor such as the one that is described here. We plan to go this route as opposed to simply buying ready-made ones to save on cost and also to have more ownership over the entire system we are designing. In addition to this, it will give us more practical experience designing complete systems, and provide us with a more complete idea as to how the system works a whole, which will be convenient as we try to integrate other features into this for our senior design project.

Our project will have the following design requirements:

- Be able to detect the heart rate with photo diode sensor
- Use a microcontroller to process analog signals
- Store heart rate data, and calculate heart rate (bpm)
- Display the calculated heart rate to the user

2 Block Diagram

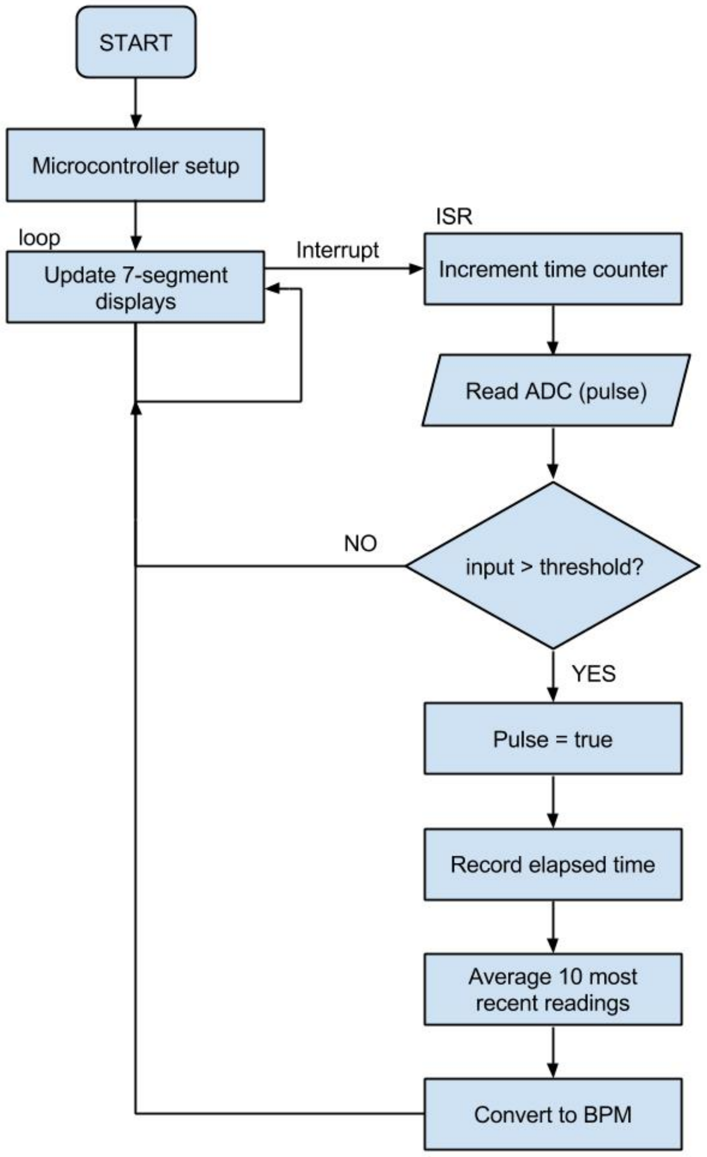
A Block Diagram that clearly shows what the project is about:



Rather than using the 7-segment displays as shown in the block diagram above, we decided to display the calculated heart rate as a 7 bit binary number (on 7 LEDs). This method still satisfies the requirement from our initial project proposal and allowed us to spend more time on the data acquisition component of the project rather than the data display. Working with 7-segment displays was completely foreign to us and would have taken much longer to get in operating order than the method we used.

3 Flow Chart

The PIC microcontroller is used as the calculating mechanism in our project, keeping track of the user's heart rate history so the more meaningful number (beats/minute) can be estimated.A comprehensive flow chart that shows the logic flow of the project's PIC processor is below.

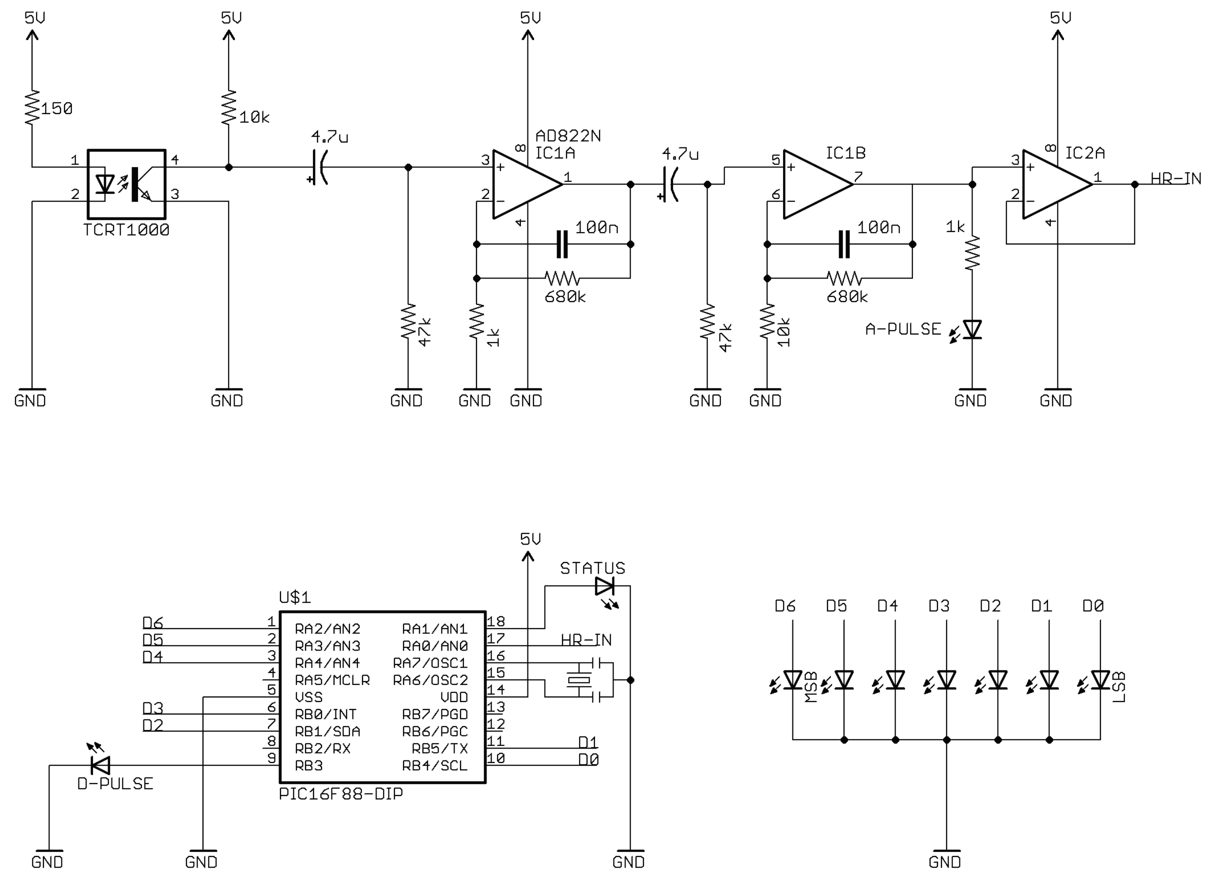


4 Evaluation

We demonstrated the functionality of the heart rate monitor we constructed to Professor Derefinko. Additionally, we recorded a video demonstration and explanation of out working Heart Rate Monitor, which can be found at: `http://youtu.be/b21cfhDZDGo`.

Listing 1: HRM Bill of Materials

```
Part     Value         Device           Description
A-PULSE                LED3MM           LED
C1       100n          CAPKIT           Capacitor
C2       100n          CAPKIT           Capacitor
C3       4.7u          CAP_POLPTH2      Capacitor Polarized
C4       4.7u          CAP_POLPTH2      Capacitor Polarized
D-PULSE                LED3MM           LED
D1                     LED3MM           LED
D2                     LED3MM           LED
D3                     LED3MM           LED
D4                     LED3MM           LED
D5                     LED3MM           LED
IC1      AD822N        AD822N           Single-Supply, Rail-to-Rail Op Amp
IC2      AD822N        AD822N           Single-Supply, Rail-to-Rail Op Amp
LSB                    LED3MM           LED
MSB                    LED3MM           LED
OK1      TCRT100       TCRT100          Optocoupler, Phototransistor Output
R1       47k           RESISTOR-1/4W    Resistor
R2       1k            RESISTOR-1/4W    Resistor
R3       680k          RESISTOR-1/4W    Resistor
R4       47k           RESISTOR-1/4W    Resistor
R5       10k           RESISTOR-1/4W    Resistor
R6       680k          RESISTOR-1/4W    Resistor
R7       10k           RESISTOR-1/4W    Resistor
R8       150           RESISTOR-1/4W    Resistor
R9       1k            RESISTOR-1/4W    Resistor
STATUS                 LED3MM           LED
U$1      PIC16F88-DIP  PIC16F88-DIP     8-bit PIC uC
Y1                     RESONATORPTH     Resonator
```



TITLE: PHR_revA

Design by:
Eddie Samuels & Jeremy Warner

REV:

Date: 12/13/14 5:40 PM

Sheet: 1/1

Listing 2: HRM Code

```
#include <pic.h>

// Definitions
#define HRIN RA0  // Signal in
#define HROUT RB2 // Pulse
#define STAT RA1  // Status
#define LED0 RB4  // LSB
#define LED1 RB5
#define LED2 RB1
#define LED3 RB0
#define LED4 RA4
#define LED5 RA3
#define LED6 RA2 // MSB



// macros for a C bit array
#define setBit(A,k)     ( A[(k/32)] |= (1 << (k%32)) )
#define clearBit(A,k)   ( A[(k/32)] &= ~(1 << (k%32)) )
#define getBit(A,k)     ( A[(k/32)] & (1 << (k%32)) )

// Variables
unsigned int threshold = 191;
int hrHistory[2];
int histSize = 64;
int frameScale = 5;
int hrLoc = 0;
int hrPulses;
int hrBPM;

// parsing ADC input
unsigned int readADC(void) {
    GO_DONE = 1;
    while (GO_DONE); // wait until conversion done
    return ADRESH;   // get and return stored ADC val (upper 8 bits of 10)
}

// delay for testing
void delay(unsigned int amount){
    unsigned int index;
    for (index = 0; index < amount; index++);
}

// looping section of code
void loop(){

    // get current value of signal
    unsigned int hrSig = readADC();
```

```
// move signal into array, shift
if(hrSig > threshold){
            setBit(hrHistory, hrLoc); HROUT = 1;
    } else {
            clearBit(hrHistory, hrLoc);  HROUT = 0;
    }

// look for pulse detections in array
hrPulses = 0;

// increment HR count - rising edge detect
for(int i = 1; i<histSize; i++ ){
    // if cur hist value high, and last val low, increment
    if(getBit(hrHistory,i) && (!getBit(hrHistory,i-1))) hrPulses++;
}

// scale frame HR to BPM
hrBPM = hrPulses * frameScale;

    // turn status LED on (RB3), if weird HR
    STAT = ( hrBPM > 220 | hrBPM < 30 )? 1 : 0;

    // print out BPM - binary to LEDs
    for (int c = 6; c >= 0; c--) {

            unsigned int k = hrBPM >> c;
            unsigned int status = k & 1;

            if (c == 6){
                    LED6 = k;
            }else if (c == 5){
                    LED5 = k;
            }else if (c == 4){
                    LED4 = k;
            }else if (c == 3){
                    LED3 = k;
            }else if (c == 2){
                    LED2 = k;
            }else if (c == 1){
                    LED1 = k;
            }else if (c == 0){
                    LED0 = k;
            }

    }

    // check if index needs resetting
    hrLoc++; // increment value
    if(hrLoc == histSize){
```

```
                    hrLoc = 0;
          }

}

// main code, declarations, loops
void main(void) {
    CMCON  = 7;              //turn off comparator
    ADCON0 = 0b00000001; //to read from AN0
    ANSEL = 0b00011000;   //0=digital,1=analog
    TRISA = 0b00000001;   //0=output, 1=input
    TRISB = 0b00000000;   //0=output, 1=input
    while(1) loop();
}
```

5 Resources

1. Blog & schematic used as basis for project

   `http://embedded-lab.com/blog/?p=1671`

2. Updates to original blog post above

   `http://embedded-lab.com/blog/?p=5508`

3. C Code for addressing bit arrays

   `http://www.mathcs.emory.edu/~cheung/Courses/255/Syllabus/1-C-intro/bit-array.html`

4. C Code for converting a decimal int val to binary

   `http://www.programmingsimplified.com/c/source-code/c-program-convert-decimal-to-binary`

5. Information about photoplethysmogram (PPG) & sampling

   `http://pulsesensor.myshopify.com/pages/pulse-sensor-amped-arduino-v1dot1`