Heart Rate Monitor with AD8232

Brian Lopez

Goal:

- To implement the AD8232 heart rate sensor using ADC.
- Have the ATmega328P to process the incoming heart beats to provide a smoother graph.
- Send the data from ATmega to an HC-05 Bluetooth module that is connected to a PC.
- Use a serial communication data processor to plot the incoming data received from the HC-05.

Deliverables:

The project will allow people to see their pulse in real time. It utilizes 3 sensor pads that are connected to a person's body and will have the value be read by the ATmega328P using the ADC interface. The ATmega will then smooth out the signal and then send the data into a HC-05 Bluetooth module. The HC-05 will be connected to a PC that will receive the data and make a real-time graph to show the person's heartbeat on the screen.

I. Literature survey

Being able to view a person's heart rate is important to be able to detect any heart problems that are hard to detect. A visual representation of a heart rate can be used to see a person's beats per minute (BPM), which can be used to detect any anomalies with the heart. Some of the conditions that can be found with just the BPM are Tachycardia, which is a very fast heart rate, Bradycardia, a slow heart rate, and many other sorts of Arrhythmias, which is just an unusual pattern with a person's heart rate. Being able to detect these conditions is important to be able to find and prevent any other problems that happen as a result.

II. Components

A. AD8232

The AD8232 is used for electrocardiography (ECG), which simply means it records the electrical activity of the heart. It gets its input by having three biomedical sensor pads connected to the body (one on the left side of the body, one on the right, and another on the right but lower to act as a ground between the other two) that has their inputs sent through a 3.5mm jack that connects directly to the AD8232. The pads get its values by measuring the muscle movements that arise when a heart beats. Therefore, its values will not be too accurate if the subject is moving too much. The AD8232 interfaces with the ATmega328P through ADC measurements.

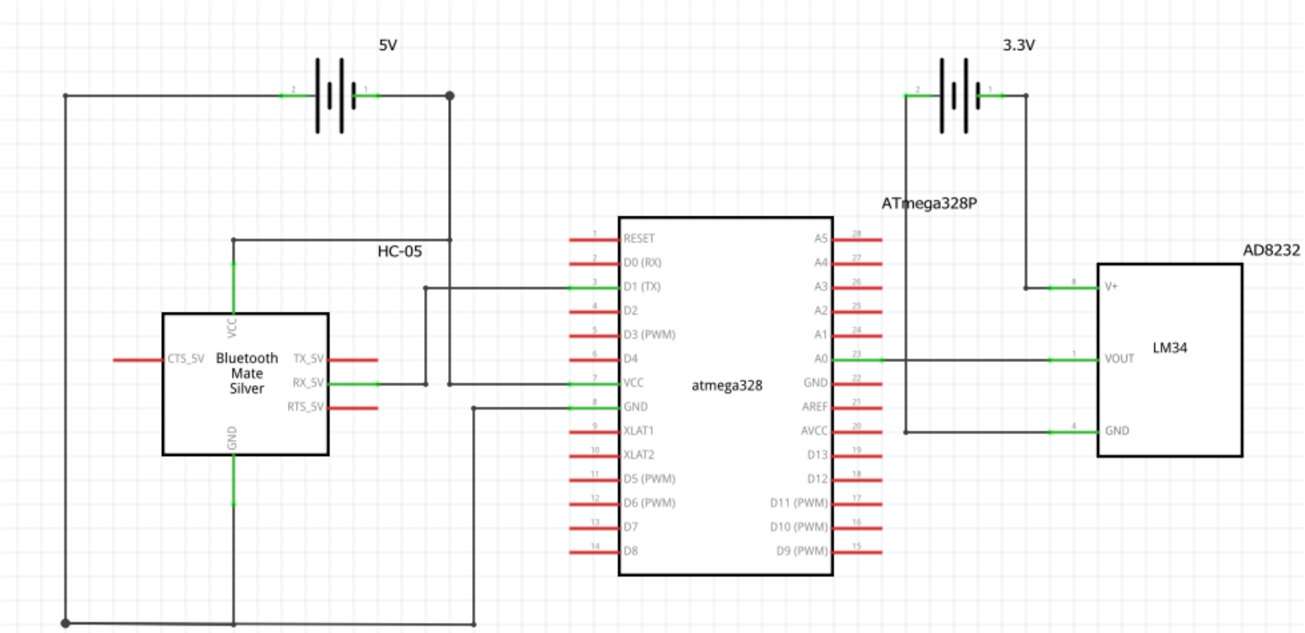
B. ATmega328P

The ATmega328P is an 8-bit microcontroller that utilizes the RISC architecture. The chip can be programmed in either AVR assembly or AVR C/C++. It consists of 8 ADC channels, 3 timers (one of which is 16 bit), SPI interface, I2C interface, 6 PWMs (only 3 true PWMs), 1 USART communication, and many more. The ATmega is programmed in Atmel Studio and is capable of plenty of inputs and outputs.

C. HC-05

The HC-05 is a Bluetooth module for wireless serial communication. It uses a 2.4GHz radio transceiver and can be used as either a master or a slave. The module uses UART communication to establish a wireless serial connection with whoever connects to the HC-05. It uses AT commands to change any settings, such as the name, baud rate, pair pin, and to set the role of the module (master or slave).

III. Schematics



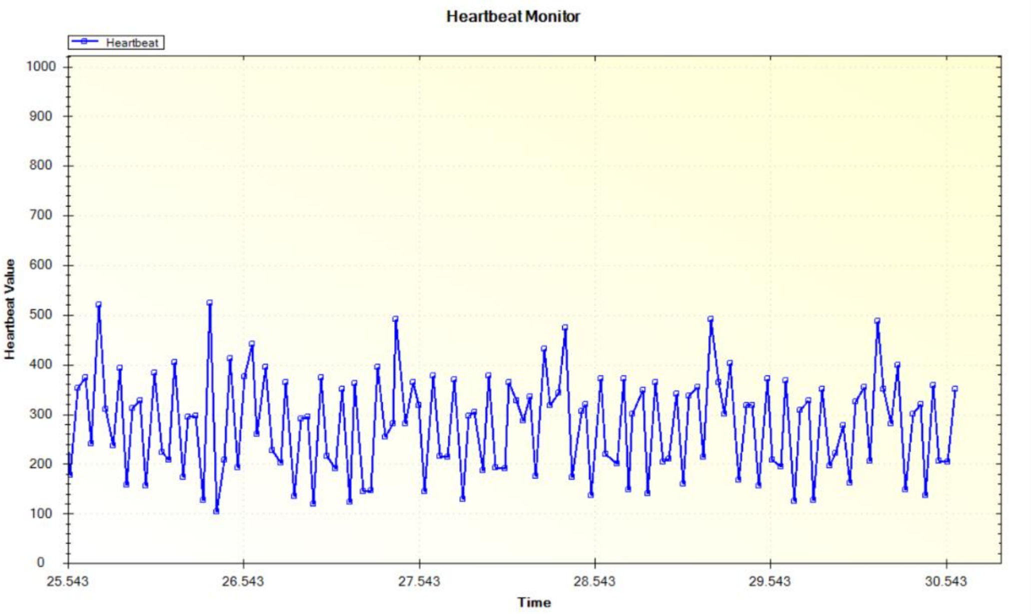
note: the LM34 pictured on the left is supposed to be an AD8232, there is no AD8232 is Fritzing.

IV. Implementation

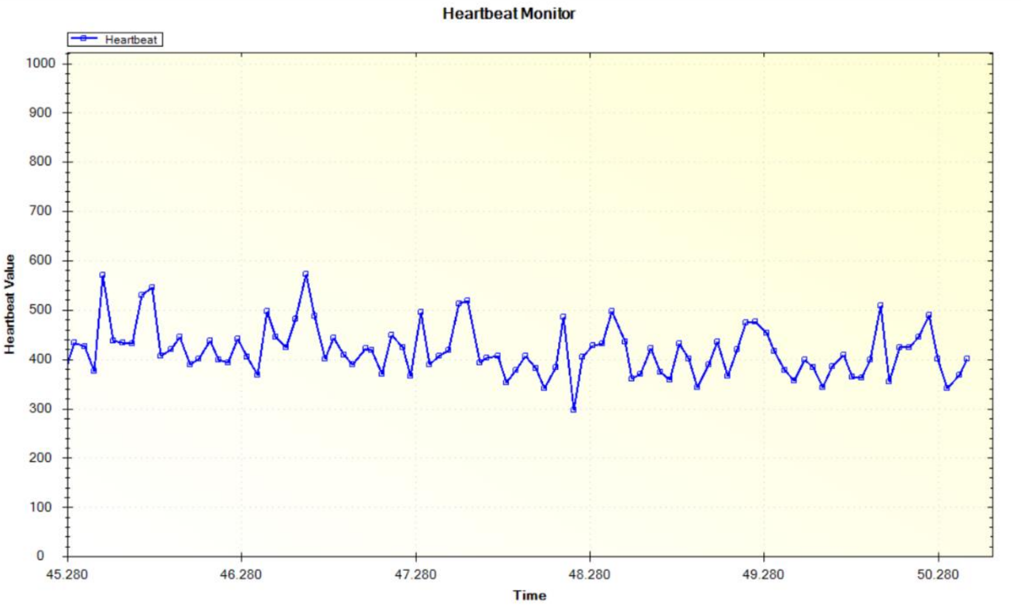
- The AD8232 is powered by a separate 3.3V power supply and will obtain its values from the sensor pads connected to a person. It will be constantly outputting its values, in which the ATmega328 will read in periodically using the ADC interface.
- Then the ATmega328 will read in the values from the AD8232 and will apply some filtering to smoothen out the ADC values. It does this by reading and summing 7 values with a 250-microsecond delay in between each one. Then it will simply find the average of those values and output that value to be sent to the HC-05 to be displayed onto the graph.
    - A FTDI was used to initially to see the values that were read by the ATmega328 before using the HC-05 to ensure that the values read are correct. Was not used in the final design however.
- The ATmega328 communicates with the PC with the HC-05 Bluetooth module. This is used to establish a wireless serial communication with each other. The HC-05 works with the ATmega328 through USART serial communication. It was configured with a baud rate of 9600 and will relay any data sent to the HC-05 towards the serial port of the PC.

V. Snapshots

With no filtering and displaying the raw input every 1ms:

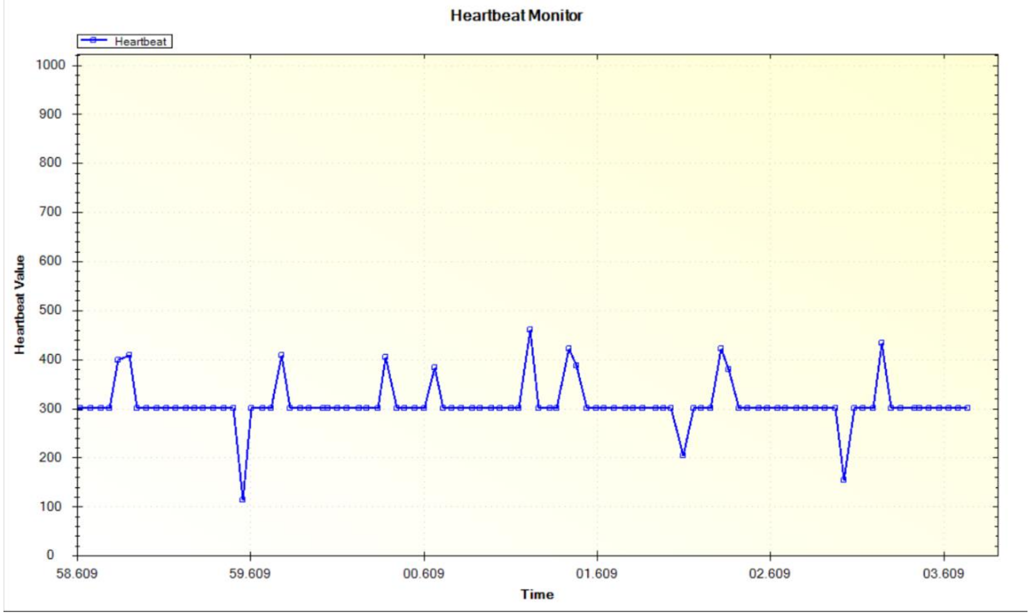


With a simple smoothing algorithm:



- The peaks are much clearer here (note that sometimes it will display two peaks right next to each other in one heartbeat. Those double peaks count as one heartbeat.).

A much more extreme smoothing algorithm was used as well that only displays the P and S peaks:



VI. Code

```
#define F_CPU 8000000UL
#include <util/delay.h>
#include <stdio.h>
#include <stdlib.h>
#include <avr/io.h>
#include <avr/interrupt.h>

void init_uart(){
  // setting the baud rate for 9600
  UBRR0H =0x00;
  UBRR0L =0x0C;
  // enabling TX & RX
  UCSR0B = (1<<RXEN0)|(1<<TXEN0);              // enable receive and transmit
  UCSR0A = (1<<UDRE0)|(1<<U2X0);
  UCSR0C =  (1 << UCSZ01) | (1 << UCSZ00);    // Set frame: 8data, 1 stop

}

void ADC_init() {
  ADMUX = 0;                 // read from port ADC0
  ADMUX |= (1<<REFS0);       // use AVcc for reference
  ADCSRA |= (1<<ADPS2) | (1<<ADPS1); // prescalar of 64
  ADCSRA |= (1<<ADEN);       // enable ADC
  ADCSRB = 0;                // free running mode
}

void USART_Transmit( char *data)
{
  while((*data != '\0')) {    // transmits all chars but null
    while(!(UCSR0A & (1<<UDRE0)));  // waits for transmit flag to clear
    UDR0 = *data;           // transmit next char
    data++;                 // move to next char
  }
}

unsigned int readADC()
{
  ADMUX &= ~(1<<ADLAR);              // clear the adc value
  unsigned int val = 0;
  ADCSRA |= (1 << ADSC);             // start adc
  while(ADCSRA & (1<<ADSC));         // wait until adc is done

  val = ADC;                    // obtain the ADC value

  return val;
}

int main(void)
{
  int beat;        // will contain the heart beat value that will be plotted
  char outs[40];   // string that will be transmitted the BT module
  init_uart();     // initialize the UART
  ADC_init();      // initialize ADC

  _delay_ms(250);

  // setting up the graph for meguno-link time graph.
  USART_Transmit("{TIMEPLOT|SET|title=Heartbeat Monitor}\n");  // set title
  USART_Transmit("{TIMEPLOT|SET|y-label=Heartbeat Value}\n");  // set y-value
  USART_Transmit("{TIMEPLOT|STYLE|Heartbeat:bs_2}\n");         // label the values

  while(1)
  {
    // simple smoothing algorithm
    // will read in seven ADC values with a small delay after each one, then
    // it will find the average of the values for a smoother value.
    for(int i = 0; i < 7; i++)
    {
      beat += readADC();
      _delay_us(250);
    }

    beat = beat / 7;     // find the average of the ADC values

    /*
    Used for "Super smoothing" the heart beat. This will only the R and S
    peaks, which can be used just to see the beats. It will just make all of
    the middle values the same constant value so that it shows a straight line
    on the non-peak values.

    if (beat < 380 && beat > 210)
      num = 300;
    */

    // Get the string for displaying the point to the graph.
    snprintf(outs, sizeof(outs), "{TIMEPLOT|DATA|Heartbeat|T|%d}\n", beat);
    USART_Transmit(outs);   // transmit it to the BT module
    beat = 0;       // reset the heart beat value.
  }

  return 0;
}
```

VII. Links

Video Demonstration: https://www.youtube.com/watch?v=1x0dk_k_yhs
Presentation Video: https://www.youtube.com/watch?v=QpQ4U7X2_oA&feature=youtu.be

References

AD8232 Datasheet: http://www.analog.com/media/en/technical-documentation/data-sheets/ad8232.pdf
ATmega328P Datasheet: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf
HC-05 Datasheet: http://www.electronicaestudio.com/docs/istd016A.pdf
Heart BeatConditions:http://www.heart.org/HEARTORG/Conditions/Arrhythmia/AboutArrhythmia/About-Arrhythmia_UCM_002010_Article.jsp