Title: Heart Rate Monitor using Pulse Sensor

By: Vincent Tuason

**Goal:**

The goal of this project is to display the heartbeat signal in its entirety using the Pulse Sensor. I will accomplish this by collecting and displaying data recorded by the Pulse sensor onto the UART terminal (data visualizer). Once I can output a steady pulse I will implement a heartbeat counter to calculate how many times the heart beats per minute. I will then use a HC06 serial Bluetooth brick to transmit all of that information on a separate terminal via Bluetooth. (I did not have an android phone, so I had to use my laptop which had Bluetooth capability)

**Deliverables:**

**This project will use a Pulse Sensor to monitor an individual’s heart rate. While each raw pulse is detected and recorded, we will also count how many times that individual’s heart beat in a span of one minute. We will then transmit this information (raw pulse graph and Beats per minute) to another terminal via Bluetooth.**

# **Literature survey[[1]](#footnote-1)**

# **Components**

## ATMega328P Xplained Mini

The ATMega328P Xplained mini is a hardware platform for interfacing with the ATMega328P microcontroller. The ATMega328P is a low power CMOS 8-bit microcontroller that uses the modified AVR RISC architecture. This microcontroller features Advanced RISC architecture, high endurance non-volatile memory segments, 8 and 16-bit timers, 8 and 6 channel 10-bit ADCs, a programmable serial USART and more. To program this chip, you can either use Assembly or C programming languages. This board comes with a built in ATMega328P microcontroller as well as an integrated debugger that could be utilized using Atmel Studio 7. The Xplained Mini features a variety of resources that can be used by the programmer such as a 16MHz target clock, target SPI bus header foot print, and an auto ID for board identification on target MCU.

## HC06 Serial Bluetooth Brick

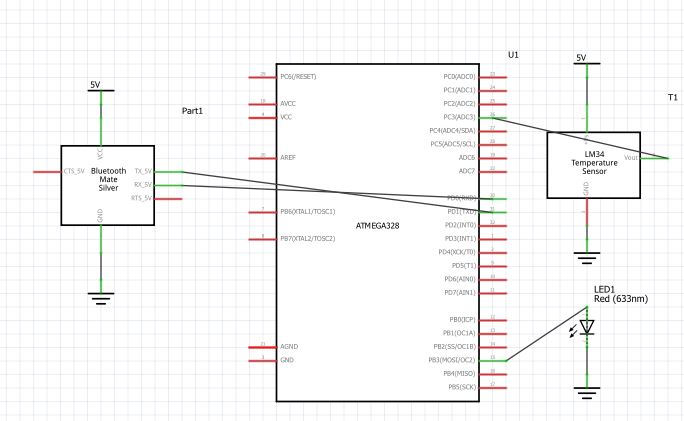
The HC06 Serial Bluetooth Brick is a plug and play electric component that can be connected to hardware UART or analog UART on the control board. You can then use it to perform wireless transmissions via Bluetooth connection (2.4 GHz) from one machine to the Bluetooth module.

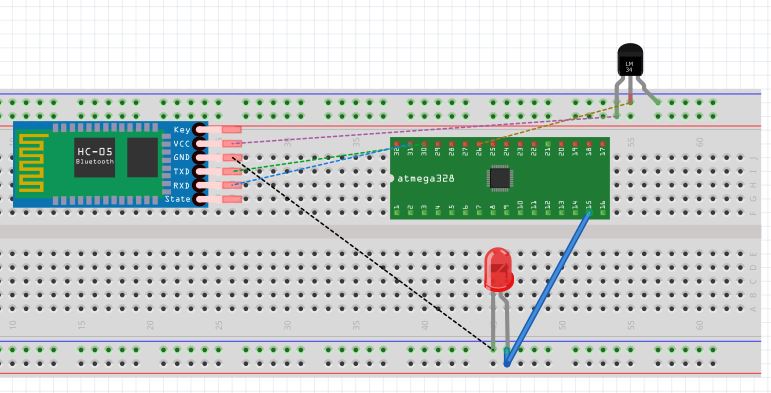
## Pulse Sensor

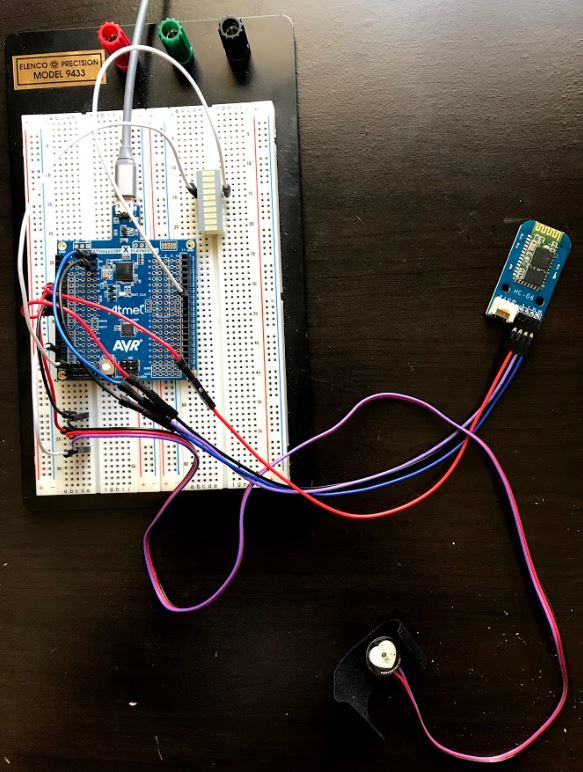
The Pulse Sensor is a low-cost optical heart rate sensor for any microcontroller with an ADC. It has three wires with the colors black, red and purple. Each of these wires go to GND, VDD which can either be 5 or 3.3 V, and the Pulse signal respectively.

# **Schematics**

Note: I could not find the Pulse Sensor symbol on fritzing so I just used an LM34 sensor as a filler since it had 3 connections. I was also not able to find the HC06 Bluetooth module on fritzing so I had to settle for an HC05 module. The first image is the schematic generated by fritzing, the second image is the breadboard implementation generated by fritzing, and the third image is my implementation of the circuit.







# **Implementation**

* Bluetooth module (HC-06) paired with the Bluetooth of PC (laptop).
* The ADC is set up to where it will read an analog signal in PC3 which is the same pin where the pulse signal will go.
* Every second, we will execute an overflow ISR and will read an ADC value. If this value is greater than the threshold value, then it is a legitimate beat thus the ADC value/raw pulse is recorded. Every time the system finds a legitimate pulse, the beat counter is incremented the LED connected to Port B will flash.
* The program will keep polling for a pulse until the elapsed time variable hits 60 which represents 60 seconds has passed. The BPM (Beats Per Minute) will then be printed on the UART terminal.
* Each pulse signal and BPM value will then be transmitted via Bluetooth on another window of Atmel Studio. The demonstration video will also show that the only thing connected to the UART terminal of the second Atmel Studio window is the Bluetooth module and nothing else.

# **Snapshots/SCREENSHOTS/VIDEOS**

(only links - do not embed images or videos in the document)

Project Demonstration Video Link: <https://www.youtube.com/watch?v=ZeoMACo3YGY>

PowerPoint Presentation with voice: <https://www.youtube.com/watch?v=lSMToQT1ECA&t=1s>

**CODE:**

#define *F\_CPU* 16000000UL //set clock at 16MHz

#define BAUD 9600 //set baud rate of 9600

#define MYUBRR *F\_CPU*/16/BAUD-1

#include <avr/io.h> //include necessary libraries

#include <avr/interrupt.h>

#include <stdio.h>

#include <util/delay.h>

void adc\_init(void); //declare a void function that will initialize the ADC

void read\_adc(void); //declare a void function that will read the ADC

void USART\_tx\_string(char\*data); //declare a void function that will print the raw pulse and BPM

void USART\_init(unsigned int ubrr); //declare a void function that will initialize USART

volatile unsigned int heart\_rate; //declare a changing variable that will hold the heart rate value

volatile unsigned int BeatsPerMinute;

volatile unsigned int timeElapsed;

char outs[20]; //declare an array of characters with size 20

char printMe[20];

int Threshold = 400; //variable to compare the value read from the sensor

int timeVariable = 60; //variable to hold 60 seconds

int main(void)

{

DDRB = 0xFF;

adc\_init(); //call the adc\_init function to initialize the ADC

USART\_init(MYUBRR); //call the USART\_init function and pass MYUBRR to it

USART\_tx\_string("Reading:\r\n"); //pass the string "Reading:" and print it on the visualizer

*\_delay\_ms*(125); //wait for 125 ms

sei(); //set interrupt enable, every time timer1 overflows, we execute the ISR

while(1)

{

}

}

void adc\_init(void) //this function is responsible for setting up and enabling the ADC

{

/\*

The line below sets each bit of the ADMUX register.

(0<<REFS1) and (1<<REFS0) sets the refrence voltage. Uses AVcc with external capacitor at AREF pin.

(0<<ADLAR) is to shift the ADC result to the right

(1<<MUX1) | (1<<MUX0) sets the single ended input as ADC3. connect the middle pin of the LM34 to PC3

\*/

ADMUX = (0 << REFS1) | (1<<REFS0) | (0<<ADLAR) | (1<<MUX1) | (1<<MUX0);

/\*

The line below sets each bit of the ADSCRA register

(1<<ADEN) is for enabling the ADC

(0<<ADSC) ADC start conversion

(0<<ADATE) Disable auto triggering of the ADC

(0<<ADIF) set ADC interrupt flag to 0

(0<<ADIE) set ADC interrupt enable to 0

(1<<ADPS2) (0<<ADPS1) (1<<ADPS0) set ADC prescaler to 32

\*/

ADCSRA = (1<<ADEN) | (0<<ADSC) | (0 << ADATE) | (0<<ADIF) | (0<<ADIE) | (1<<ADPS2)| (0<<ADPS1) | (1<<ADPS0);

TIMSK1 |= (1<<TOIE1); //enable Timer1 overflow interrupt

TCCR1B |= (1<<CS12) | (1<<CS10); //set timer prescale to 1024

TCNT1 = 49911; //set TCNT value. Calculated using 65535 - (16MHz/1024-1)

}

void read\_adc(void) //this function is responsible for reading the ADC pins

{

unsigned char i=4; //declare how many times we will measure raw pulse per average

heart\_rate=0; //initialize heart\_rate to 0

while(i--) //while i != 0, keep looping

{

ADCSRA |= (1<<ADSC); //start the first conversion

while(ADCSRA & (1<<ADSC)); //while ADCSRA and 1 is written to ADSC, keep looping

heart\_rate+=ADC; //sum the ADC values

*\_delay\_ms*(50); //wait for 50 ms

}

heart\_rate = heart\_rate /4; //take the average of the measured pulse sensor values

}

void USART\_init(unsigned int ubrr) //this function is responsible for initializing USART (RS-232)

{

UBRR0L = (unsigned char)ubrr; //this line and the next line sets a baud rate of 9600

UBRR0H = (unsigned char)(ubrr>>8);

UCSR0B = (1<<TXEN0); //enable the USART Transmitter

UCSR0C = (3<<UCSZ00); //sets the number of data bits in a frame the receiver and transmitter use

}

void USART\_tx\_string(char\*data) //function responsible for printing the raw pulse and BPM

{

while((\*data != '\0')) //check if the character being read is = '\0' if it is equal stop looping

{

while(!(UCSR0A & (1<<UDRE0)));

UDR0 = \*data;

data++; //increment data

}

}

ISR(TIMER1\_OVF\_vect)

{

timeElapsed = timeElapsed + 1;

read\_adc(); //get the pulse

if(heart\_rate > Threshold) //If the ADC value that was read is greater than the threshold, print the raw pulse

{

BeatsPerMinute = BeatsPerMinute + 1; //increment the Heart Beat counter

*snprintf*(outs,sizeof(outs), "%3d Raw Pulse\r\n", heart\_rate); //store the raw pulse into a string of 20 characters

USART\_tx\_string(outs); //print the pulse

PORTB = 0xFF; //turn on LED every time a raw pulse is recorded

*\_delay\_ms*(50); //wait for 50 ms

PORTB = 0; //turn off the LED

TCNT1 = 49911; //reset the timer for next iteration

if(timeElapsed == timeVariable) //if the timeElapsed is equal to 60 seconds, print the Number of Beats

{

*snprintf*(printMe,sizeof(printMe), "%3d BPM\r\n", BeatsPerMinute); //store the raw pulse into a string of 20 characters

USART\_tx\_string(printMe); //print the BPM

timeElapsed = 0; //reset the time elapsed

BeatsPerMinute = 0; //reset the number of beats

}

}

//If the ADC value that was read is less than the threshold, the pulse is not legitimate. Most likely due to noise.

else

{

TCNT1 = 49911; //reset the timer for next iteration

}

}

# Links

GITHUB Link:

<https://github.com/vason13/vasonGIT.git>

# Conclusion

Please include a brief summary of the possible clinical implications of your work in the conclusion section. Although a conclusion may review the main points of the paper, do not replicate the abstract as the conclusion. Consider elaborating on the translational importance of the work or suggest applications and extensions.

References

**ATMega328P Complete Datasheet:**

<http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf>

**Pulse Sensor Datasheet:**

<https://cdn.shopify.com/s/files/1/0100/6632/files/Pulse_Sensor_Data_Sheet.pdf?14358792549038671331>

**HC06 Serial Bluetooth Brick Datasheet:**

<ftp://imall.iteadstudio.com/Electronic_Brick/IM120710006/DS_IM120710006.pdf>

**LED Block:**

<http://www.farnell.com/datasheets/1683489.pdf?_ga=2.170816608.1716658936.1525797591-1462882040.1525797591&_gac=1.242035382.1525797591.EAIaIQobChMI4sbl9sb22gIVhUOGCh0yYQJGEAAYAiAAEgK9j_D_BwE>

1. [↑](#footnote-ref-1)