Stefan Prestrelski
Belyea
Embedded Systems Period 2
6/04/2018
Final Project Report

Heartrate Sensor

1. Introduction

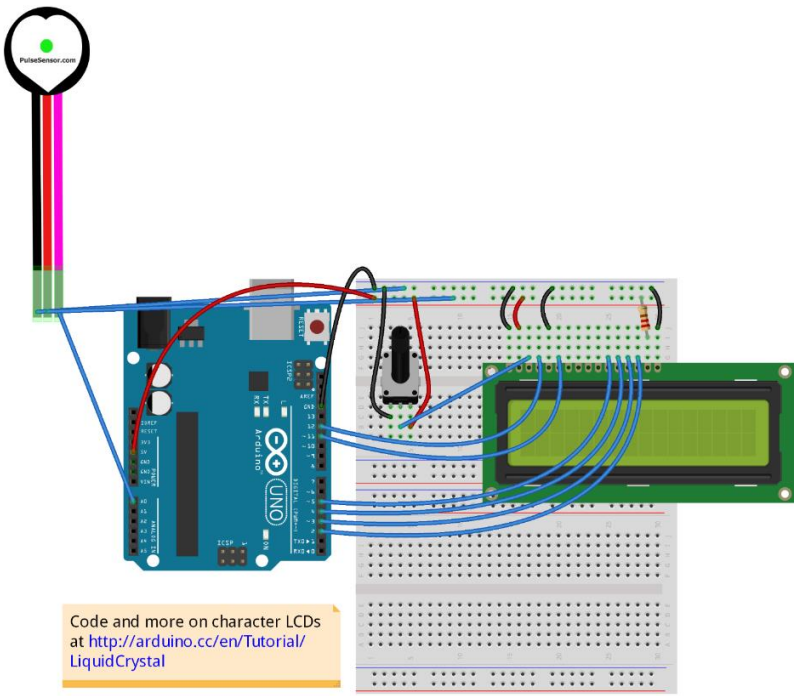
The heartbeat sensor is designed to take a pulse and measure heartbeat in beats per minute. How it works is detecting the shockwaves produced by the heart beating, and then measuring the intensity and frequency of them. In this project, I used an Arduino with the aforementioned heartbeat sensor, and an LCD display to display results. My project is a modification of an existing design from https://github.com/WorldFamousElectronics/PulseSensorPlayground. The source also provides code for the heartrate sensor, which I modified quite a bit to display the results on an LCD screen using C++. I also had to integrate a library from the following website to use the LCD screen: https://www.arduino.cc/en/Reference/LiquidCrystal

2. Design and Materials

A schematic of my heartrate sensor and LCD display is shown below. I made the schematic using the program Fritzing http://fritzing.org/home/

Materials:

- Arduino Uno (made by Sunfounder)
- Breadboard
- 10K ohm potentiometer
- 16x2 LCD display (made by Sunfounder)
- Heartrate sensor (made by Sparkfun Electronics)
- Various jumper wires
- Resistor (220 ohm)



3. Computer Code

Below is the code for the Arduino in C++, which can be found here on my Github account: https://github.com/StefPres/Heartbeat_Sensor-LCD

This Code uses Libraries from the following sources to be able to access the functionality of the heartrate sensor and the LCD Display:

https://github.com/WorldFamousElectronics/PulseSensorPlayground

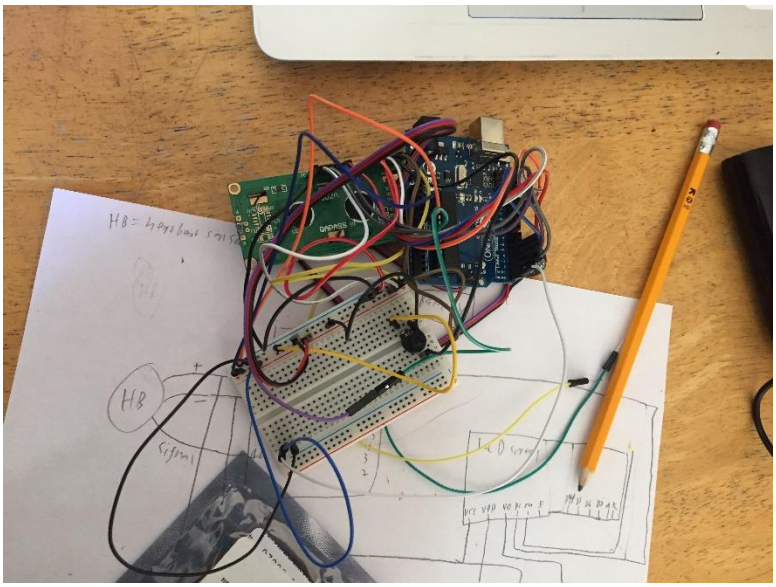
https://www.arduino.cc/en/Reference/LiquidCrystal

Code:

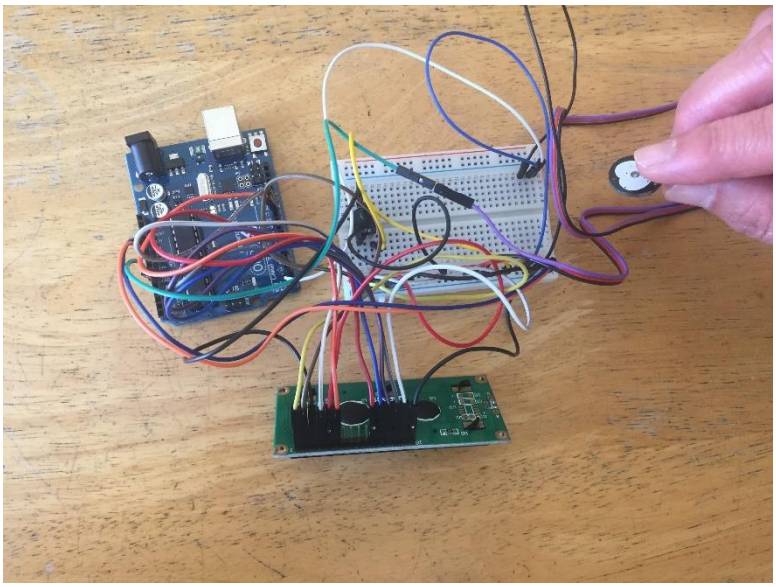
```
#define USE_ARDUINO_INTERRUPTS true
#include <PulseSensorPlayground.h>
#include <LiquidCrystal.h>

//  Variables
const int PulseWire = 0;
const int LED13 = 13;
int Threshold = 550;

PulseSensorPlayground pulseSensor;
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  Serial.begin(9600);
```

```
  pulseSensor.analogInput(PulseWire);
  pulseSensor.blinkOnPulse(LED13);
  pulseSensor.setThreshold(Threshold);
   if (pulseSensor.begin()) {
    Serial.println("We created a pulseSensor Object !");
  }
  lcd.begin(16, 2);
}

void loop() {

 int myBPM = pulseSensor.getBeatsPerMinute();

if (pulseSensor.sawStartOfBeat()) {
 Serial.println("❤  A HeartBeat Happened ! ");
 Serial.print("BPM: ");
 Serial.println(myBPM);
 lcd.setCursor(1, 0);
 lcd.print("Heartbeat!");
 lcd.setCursor(1, 1);
 lcd.print("BPM: ");
 lcd.print(myBPM);
}
  delay(20);

}
```

Here is a photo of the heartbeat sensor in progress:



4. Results:



I performed a test on 5 students in the Embedded Systems class with three trials per student. I started by calculating the pulse using a stopwatch, and then performed the test using the heartrate sensor. The results are displayed below.

Test Subject (students from Embedded Sys Class)	Trial 1	Trial 2	Trial 3	Average/10	Pulse measured using stop watch
Daniel Liu	730	740	730	73.3	74
Dennis Li	750	740	760	75.0	75
Eric Oh	720	730	710	72.0	73
Fangli	750	750	730	74.6	76
Gil	730	780	710	74.0	74

As shown above, I will have to add a conversion factor of 10. After 3 trials, the results are all very close to each other, and dividing the average by 10 gives a result very close to the true value. Overall, the sensor is quite accurate, and quite precise as well.

5. Discussion and Conclusion

It looks as though there is a conversion issue with this particular sensor on a factor of 10 from time between beats to beats per minute. However, dividing the results by 10 gives a result very close to the true value, and it is likely that I may need to adapt the code to compensate for the error (probably divide the results by a factor of 10). Adapting the system to a commercial-grade heartrate sensor might give more accurate results.