Building and Visualizing Datasets of a Single Variable (Heart Rate)

using Arduino, XD-58C and Processing

Neel Adwani
University of Petroleum and Energy Studies
Dehradun, Uttarakhand, India

Abstract: Datasets are of great use in the fields like Machine Learning, Data Science, Data Analytics, etc. To solve the problem of Data unavailability, Data sets can be generated with the help of required sensors and that data can even be visualized. Visualization might be necessary to filter or sort out the inaccurate data, which might interfere with the Algorithm used.

Introduction:

To get the input for building the dataset, An XD-58C heart rate sensor, an Arduino Uno are used. On the Software side, Arduino IDE and Processing are used. Here's the description of all the components:

Hardware Components:

1. **Arduino UNO:** It is an open-source development board based on ATmega328P microcontroller. It is developed by arduino.cc.
2. **XD-58C Heart Rate Sensor:** It is a sensor used for measuring the heart rate. It constitutes of an LED and a circuit. It works on the principle that when the blood flows inside the veins, the light coming from the LED gets reflected, that is how it can detect the heart rate.

Software Components:

1. **Arduino IDE:** It is an open-source integrated development environment created by arduino.cc, which is specifically built for programming Arduino boards.
2. **Processing:** It is an open-source graphical library and an integrated development environment, basically implemented in Java.

Hardware Setup:

1. 5V of the Arduino UNO is connected to the positive of XD-58C.
2. GND of the Arduino UNO is connected to the negative of the XD-58C sensor.
3. A0(Analog Pin) of the Arduino UNO is connected to 'S' of the XD-58C sensor.
4. The Arduino should be connected to the PC via an A/B cable for Arduino.

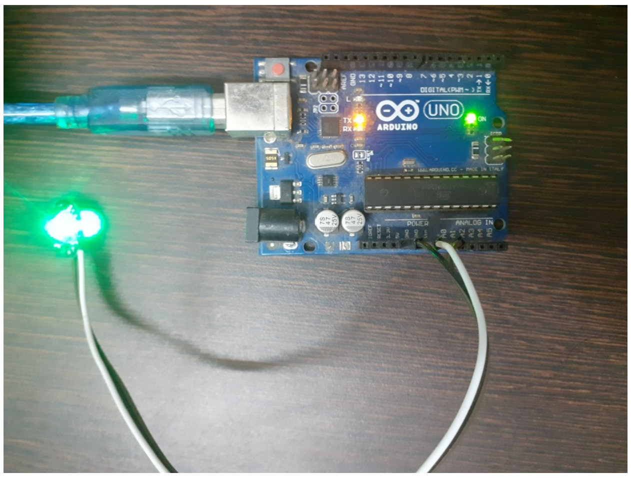


Figure 1: Arduino UNO connected to XD-58C Heart Rate Sensor

Methodology:

Used in Arduino:

1. Import the PulseSensorPlayground library.
2. Declare variables to store the values of PulseWire, Threshold, heartbeat.
3. Create an instance of PulseSensorPlayground object.
4. Inside the setup block, start the serial monitor at 9600 bauds.
5. Inside the loop block, store the value of BPM in that variable.
6. Write the value of BPM in the serial monitor.

Used in Processing:

1. Import the processing serial library.
2. Create a variable to store the value of Serial port.
3. Declare the variables to store id, heart rate.
4. Inside the setup block, define the size of the window and create a table.
5. Add the columns for ID and Heart Rate respectively.
6. Get the list of available serial ports and store the specific one in a variable.
7. Inside the draw block, choose the stroke and fill color.
8. Draw the line in such a way that it is colored at the bottom and the boundary is present at the heart rate recorded at that moment
9. If the position of graph cursor becomes higher than that of the width, set the width equal to zero.
10. Otherwise, increment the horizontal position.
11. Create a new row, add ID and heart rate to it and save the table every time the draw block gets executed.
12. Create a function to get the serial port.
13. Inside the function, map the value of heart beat so that it fits the height accordingly.

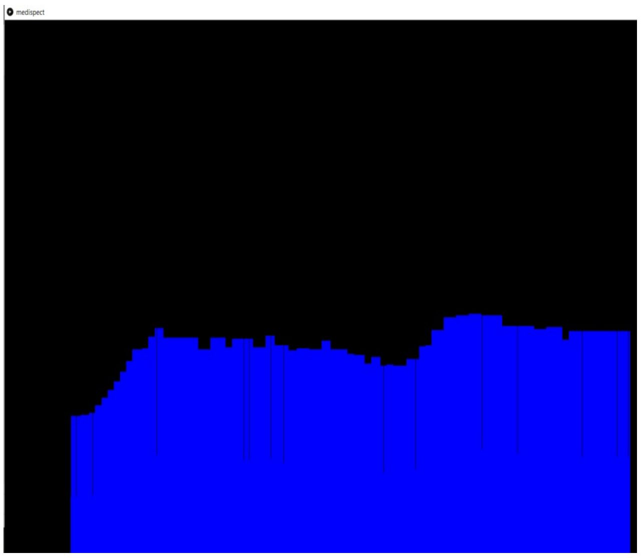
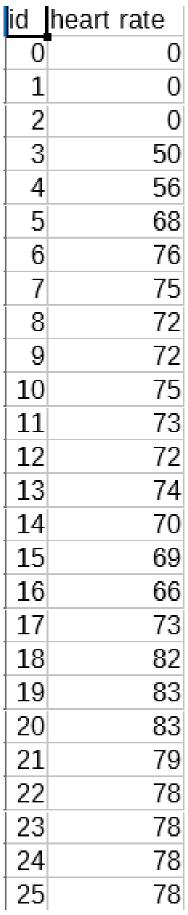


Figure 2: Heart Rate visualized



id	heart rate
0	0
1	0
2	0
3	50
4	56
5	68
6	76
7	75
8	72
9	72
10	75
11	73
12	72
13	74
14	70
15	69
16	66
17	73
18	82
19	83
20	83
21	79
22	78
23	78
24	78
25	78

Figure 3: Generated Dataset

Code:

Used in Arduino:

```
#define USE_ARDUINO_INTERRUPTS true
#include<PulseSensorPlayground.h>
int PulseWire = 0;
int Threshold = 550;
void setup()
{
Serial.begin(9600);
pulseSensor.analogInput(PulseWire);
pulseSensor.setThreshold(Threshold);
}
void loop()
{
int bpm = pulseSensor.getBeatsPerMinute();
Serial.println(bpm);
}
```

Used in Processing:

```
import processing.serial.*;
Serial myPort;int xPos = 1;
float hr = 0;
String val;
Table table;
int id;
void setup ()
{
size(1920, 1080);
table = new Table();
table.addColumn("id");
table.addColumn("heart rate");
myPort = new Serial(this, Serial.list()[0], 9600);
myPort.bufferUntil('\n');
background(0);
}
void draw ()
{
stroke(0, 0, 255);
fill(0);
line(xPos, height, xPos, height – hr);
if (xPos >= width)
{
xPos = 0;
background(0);
}
else
{
xPos++;
}
textSize(20);
fill(100);
TableRow newRow = table.addRow();
newRow.setInt("id", table.lastRowIndex());
newRow.setInt("heart rate", int(hr/6));
saveTable(table, "data/new1.csv");
}
void serialEvent (Serial myPort)
{
String inString = myPort.readStringUntil('\n');
if (inString != null)
{
```

```
inString = trim(inString);
hr = float(inString);
println(hr);
hr = map(hr, 0, 512, 0, height);
}
}
```

References:

1. G. Vega-Martinez, F. J. Ramos-Becerril, D. Mirabent-Amor, J. G. Franco-Sánchez, A. Vera-Hernández, C. Alvarado-Serrano, L. Leija-Salas, "Analysis of heart rate variability and its application in sports medicine: A review", Global Medical Engineering Physics Exchanges/Pan American Health Care Exchanges (GMEPE/PAHCE) 2018, pp. 1-5, 2018.
2. T. S. Arulananth and B. Shilpa, "Fingertip based heart beat monitoring system using embedded systems," 2017 International conference of Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, 2017, pp. 227-230.doi: 10.1109/ICECA.2017.8212802