BME 363 Biomedical Instrumentation Design Lab #6 Helper

PIC Software – Add the following code in the interrupt service routine isr()

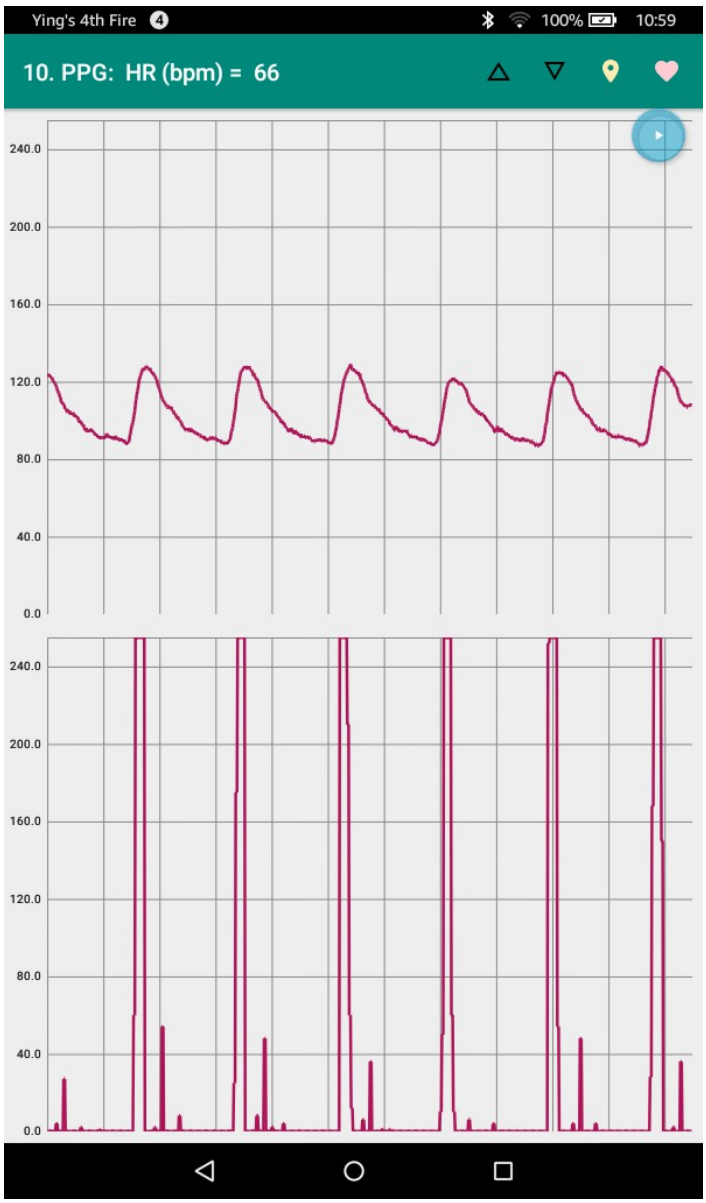
```
case 10:                        // Function 10: Photoplethysmogram
      TMR0H = 0xFE;             // Reload TMR0 high-order byte
      TMR0L = sampling_L;       // Reload TMR0 low-order byte
      PORTCbits.RC3 = !PORTCbits.RC3;     // Toggle RC3 (pin 18) @ 1 KHz for PPG
      skipCount++;
      if (skipCount == 5) {
            SetupADC(2);              // Switch to A/D channel AN2
            sampling_L = ReadADC(); // Read potentiometer setting from AN2
            SetupADC(3);              // Switch to A/D channel AN2
      }
      output = ReadADC();       // Read PPG from AN3
      d0 += output;
      if (skipCount == 8) {     // 1 KHz / 8 = 125 Hz, check pin 18 for the actual frequency
            skipCount = 0;
            d0 = d0 >> 3;       // d0 contains the sum of 8 points, then / 8 (shift 3 bits)
            output = (unsigned char) d0;
            d0 = 0;
            if (enableBT) {
                  TransmitBT(functionBT);
                  TransmitBT(output);
                  TransmitBT(128);
            }
      }
      break;
```

Android Software – Implement a 4-point MOBD algorithm. Add the following code in MainActivity.

```
Global variables ......
private static int data0, data1, d0, d1, d2, d3, rri_count;
private static int threshold=80, mobd, refractory, hr, skipCount;
private static String ppg_hr;


.............
graphValue(val1, mChartOriginal);   // Display the first channel
if (currentFunction == 10) {        // MODB for PPG pulse detection
    skipCount++;                    // Decimate sample points 846 Hz / 8 = 106 Hz
    if (skipCount >= 3) {           // Decimate by 3, 106 Hz / 3 = 35.3 Hz, or 28 ms
        skipCount = 0;              // Reset skipCount
        data1 = data0;              // Move old PPG sample to data1
        data0 = val1;               // Store new PPG sample from ADC to data0
        d3 = d2;                    // Move oldest difference to d3
        d2 = d1;                    // Move older difference to d2
        d1 = d0;                    // Move old difference to d1
        d0 = data0 - data1;         // Store new difference in d0, ( int) casting important
        rri_count++;                // Increment RR-interval
        mobd = 0;                   // mobd = 0, unless sign consistency is met:
        if (d0 > 0 && d1 > 0 && d2 > 0 && d3 > 0) {    // (1) If 4 positive differences
            mobd = d0 * d1;         // Multiply first two differences
            mobd = mobd * d2;       // Multiply the older difference
            mobd = mobd * d3;       // Multiply the oldest difference
        }
        if (d0 < 0 && d1 < 0 && d2 < 0 && d3 < 0) {    // (2) If 4 negative differences
            d0 = -d0;               // Take absolute value of differences
            d1 = -d1;
            d2 = -d2;
            d3 = -d3;
            mobd = d0 * d1;         // Multiply first two differences
            mobd = mobd * d2;       // Multiply the older difference
```

```
            mobd = mobd * d3;       // Multiply the oldest difference
        }
        if (refractory > 0) {       // Avoid detecting extraneous peaks after QRS
            refractory++;
            if (refractory == 20) {    // Delay for 560 ms
                refractory = 0;        // Reset refractory flag to 0
            }
        } else if (mobd > threshold) { // If a peak is detected,
            refractory = 1;            // Set refractory flag
            hr = 3500 / rri_count;     // Due to packet loss, this needs calibration
            if (hr > 40 && hr < 140) ppg_hr = String.format("10. PPG:  HR (bpm) = %3d", hr);
            else ppg_hr = String.format("10. PPG:  HR (bpm) = ");
            runOnUiThread(() -> setTitle(ppg_hr));  // Display HR on tool bar
            rri_count = 0;
        }
        if (mobd > 255) mobd = 255;
    }
    val2 = mobd;
}
graphValue(val2, mChartTransformed);   // Display the second channel
```



The detection is quite sensitive to the magnitude and quality of the PPG. If you encounter problems with inconsistent detections, you might want to implement an adaptive threshold as follows.

```
Global variables ......
private static int data0, data1, d0, d1, d2, d3, rri_count;
private static int threshold=20, decay, mobd, refractory, hr, skipCount;
private static String ppg_hr;


............
graphValue(val1, mChartOriginal);   // Display the first channel
if (currentFunction == 10) {        // MODB for PPG pulse detection
    skipCount++;                    // Decimate sample points 846 Hz / 8 = 106 Hz
    if (skipCount >= 3) {           // Decimate by 3, 106 Hz / 3 = 35.3 Hz, or 28 ms
        skipCount = 0;              // Reset skipCount
        data1 = data0;              // Move old PPG sample to data1
        data0 = val1;               // Store new PPG sample from ADC to data0
        d3 = d2;                    // Move oldest difference to d3
        d2 = d1;                    // Move older difference to d2
        d1 = d0;                    // Move old difference to d1
        d0 = data0 - data1;         // Store new difference in d0, ( int) casting important
        rri_count++;                // Increment RR-interval
        mobd = 0;                   // mobd = 0, unless sign consistency is met:
        if (d0 > 0 && d1 > 0 && d2 > 0 && d3 > 0) {    // (1) If 4 positive differences
            mobd = d0 * d1;         // Multiply first two differences
            mobd = mobd * d2;       // Multiply the older difference
            mobd = mobd * d3;       // Multiply the oldest difference
        }
        if (d0 < 0 && d1 < 0 && d2 < 0 && d3 < 0) {    // (2) If 4 negative differences
            d0 = -d0;               // Take absolute value of differences
            d1 = -d1;    d2 = -d2;    d3 = -d3;    // Make them all positive
            mobd = d0 * d1;         // Multiply first two differences
            mobd = mobd * d2;       // Multiply the older difference
            mobd = mobd * d3;       // Multiply the oldest difference
        }
        if (refractory > 0) {       // Avoid detecting extraneous peaks after QRS
            refractory++;
            if (refractory == 10) {     // Delay for 280 ms
                refractory = 0;         // Reset refractory flag to 0
            }
        }
        else {
            if (mobd > threshold) {     // If a peak is detected,
                refractory = 1;         // Set refractory flag
                threshold = mobd >> 1;  // Adaptive threshold
                hr = 3500 / rri_count;  // Due to packet loss, this needs calibration
                if (hr > 40 && hr < 140) ppg_hr = String.format("10. HR (bpm) = %3d", hr);
                else ppg_hr = String.format("10. HR (bpm) = ");
                runOnUiThread(() -> setTitle(ppg_hr));  // Display HR on tool bar
                rri_count = 0;          // Reset RR interval counter
                decay = 5;              // Set decay time constant
            }
            if (decay-- == 0) {         // Counting decay down to 0
                decay = 5;              // 140 ms
                threshold = threshold >> 1; // Decrease threshold by half very 140 ms
                if (threshold < 10) threshold = 10;     // Set floor for threshold
            }
        }
        if (mobd > 255) mobd = 255;
    }
    val2 = mobd;
}
graphValue(val2, mChartTransformed);   // Display the second channel
```