

```

// This Processing code has been written by Univeristy Of Cincinnati Student
Hamza Al Halabi [Alhalaha] for use in Circuits BME2081C for Summer of 2026
// No Artificial intelligence was used for this program with the unique
exception of line 32 to obtain the correct hospital monitor pitch
// This Code runs in parallel with the arduino code to provide the visual and
audible outputs
// Code is fully commented to allow for easy navigation to essential chunk
loops such as line trace, bpm tracking, sound initialization, et as well explain
certain functionalities
// Enjoy!

import processing.serial.*;
import processing.sound.*; // Import the Sound library

Serial myPort;
SinOsc beepSound; // Create the sine wave oscillator

int ecgValue = 0;

// Variables for the scrolling line
int xPos = 0;
float previousX = 0;
float previousY = 0;

// Threshold and Visual settings
int threshold = 450;
int flashTimer = 0;

// BPM Calculation Variables
boolean isPeak = false;
long lastBeatTime = 0;
float currentBPM = 0;

void setup() {
    size(1000, 500);
    background(5, 10, 5);

    // Initialize the beep sound
    beepSound = new SinOsc(this);
    beepSound.freq(700); // 700 Hz is a standard clinical monitor pitch
    beepSound.play();
    beepSound.amp(0.0); // Start it muted

    printArray(Serial.list());

    try {
        String portName = Serial.list()[12];
        myPort = new Serial(this, portName, 9600);
        myPort.bufferUntil('\n');
        println("SUCCESS: Connected to " + portName);
    } catch (Exception e) {
        println("CRITICAL ERROR: Could not connect to the COM port!");
        println("Close the Arduino Serial Monitor and try again.");
    }
}

void draw() {
    // =====
    // 1. DRAW TRACE AREA (WITH ERASE BAR & GRID)
    // =====
    float traceWidth = width - 280;

    // Erase the path ahead of the drawing line
    noStroke();

```

```

fill(5, 10, 5);
rect(xPos, 0, 40, height);

// Draw faint clinical grid lines
stroke(15, 30, 15);
strokeWeight(1);
for (int i = 0; i < height; i += 50) {
  line(xPos, i, xPos + 40, i);
}
for (int i = 0; i < traceWidth; i += 50) {
  if (i >= xPos && i < xPos + 40) {
    line(i, 0, i, height);
  }
}

// =====
// 2. THRESHOLD, BPM, AND AUDIO LOGIC
// =====
if (ecgValue >= threshold && !isPeak) {
  flashTimer = 10;
  isPeak = true;
  beepSound.amp(0.4); // Turn the beep volume up when threshold is crossed

  long currentTime = millis();
  long timeInterval = currentTime - lastBeatTime;

  // Filter noise: Calculate only if interval is between 300ms (200 BPM) and
2000ms (30 BPM)
  if (timeInterval > 300 && timeInterval < 2000) {
    currentBPM = 60000.0 / timeInterval;
  }
  lastBeatTime = currentTime;
}

if (ecgValue < threshold - 20) {
  isPeak = false; // Reset lock
}

// =====
// 3. DRAW RIGHT-SIDE VITALS PANEL
// =====
noStroke();
fill(15, 20, 15);
rect(traceWidth, 0, 280, height);

// Panel Border Line
stroke(30, 60, 30);
strokeWeight(3);
line(traceWidth, 0, traceWidth, height);

// Beating Heart Icon & Audio Mute Logic
if (flashTimer > 0) {
  fill(255, 40, 40); // Bright Red Flash
  flashTimer--;

  // Mute the beep the moment the flash finishes
  if (flashTimer == 0) {
    beepSound.amp(0.0);
  }
} else {
  fill(60, 15, 15); // Dim Red Rest
}
ellipse(traceWidth + 50, 60, 25, 25);

```

```

// Label
fill(200);
textSize(20);
text("ECG / EKG", traceWidth + 80, 67);

// MASSIVE BPM DISPLAY
fill(0, 255, 100); // Neon Green text
textSize(24);
text("HR", traceWidth + 30, 160);

textSize(80); // Giant clinical font
if (currentBPM > 0) {
  text(int(currentBPM), traceWidth + 80, 180);
} else {
  text("--", traceWidth + 80, 180);
}

textSize(20);
text("bpm", traceWidth + 210, 180);

// DYNAMIC LEVEL INDICATOR
String levelText = "ANALYZING...";
color levelColor = color(100);

if (currentBPM > 0) {
  if (currentBPM < 60) {
    levelText = "LOW (BRADYCARDIA)";
    levelColor = color(255, 255, 100); // Yellow
  } else if (currentBPM <= 99) {
    levelText = "NORMAL RESTING";
    levelColor = color(100, 200, 255); // Cyan
  } else if (currentBPM <= 139) {
    levelText = "ACTIVE";
    levelColor = color(255, 150, 0); // Orange
  } else {
    levelText = "ELEVATED";
    levelColor = color(255, 50, 50); // Red
  }
}

fill(levelColor);
textSize(22);
text(levelText, traceWidth + 30, 240);

// Debug/Raw Data at the bottom of the panel
fill(0, 150, 50);
textSize(14);
text("RAW SIGNAL: " + ecgValue, traceWidth + 30, height - 50);
text("THRESHOLD: " + threshold, traceWidth + 30, height - 30);

// =====
// 4. DRAW THE ECG TRACE
// =====
float mappedY = map(ecgValue, 300, 600, height - 40, 40);

stroke(0, 255, 50); // Bright Hospital Green
strokeWeight(3);

if (xPos > 0 && xPos < traceWidth) {
  line(previousX, previousY, xPos, mappedY);
}

previousX = xPos;
previousY = mappedY;

```

```

    xPos += 3; // Speed of the scan line

    // Wrap around when it hits the panel
    if (xPos >= traceWidth) {
        xPos = 0;
        previousX = 0;
    }
}

// Automatically triggers whenever Arduino sends data
void serialEvent(Serial myPort) {
    String inString = myPort.readStringUntil('\n');
    if (inString != null) {
        inString = trim(inString);
        try {
            ecgValue = int(inString);
        } catch (NumberFormatException e) {}
    }
}

```