Setting Up Node Server, iPhone Motion Streaming & Garmin HR in p5.js

Technical Documentation & Full Code (for submission)

1. Original Prompt (asked inside ChatGPT)

“Can I connect a Bluetooth chest heart rate monitor up to a p5.js sketch? What about Garmin or Apple Watch? What other stats can I extract? Using my phone for movement works well, but can we transfer this data over network to my laptop? So phone is used alongside heart rate from strap or watch and both feed into a sketch running on my laptop.”

2. System Overview

With ChatGPT’s guidance, I built a complete pipeline that:
• Streams **accelerometer motion** from an iPhone to a Node.js server over WebSockets
• Retrieves **motion data** in the p5.js laptop sketch via `/motion`
• Connects directly from the browser to a **Garmin HR strap** via Web Bluetooth
• Displays heart rate + motion values, and drives a generative animation

This PDF documents the full, working version of: server.js, phone.html, and sketch.js.

3. Node.js Server (server.js)

This server provides:
• HTTPS hosting for p5 sketch and phone control page
• A WebSocket for streaming motion from the phone
• A `/motion` endpoint for the p5 sketch to poll

Below is the exact code required:

```
// server.js — HTTPS + WebSocket + Motion Buffer

const fs = require('fs');
const https = require('https');
const express = require('express');
const WebSocket = require('ws');
const app = express();

// serve public files
app.use(express.static('public'));

let latestMotion = { x: 0, y: 0, z: 0, t: 0 };

// endpoint for viewer sketch
app.get('/motion', (req, res) => {
  res.json(latestMotion);
});

// load certs
const server = https.createServer({
  key: fs.readFileSync('./cert/key.pem'),
  cert: fs.readFileSync('./cert/cert.pem'),
}, app);

const wss = new WebSocket.Server({ server });

// phone motion WS
wss.on('connection', ws => {
  console.log('[SERVER] Client connected');

  ws.on('message', msg => {
    try {
      const data = JSON.parse(msg);
      if (data.type === 'motion') {
        latestMotion = data;
      }
    } catch (err) {
      console.error('Bad WS message:', err);
    }
  });

  ws.on('close', () => console.log('[SERVER] Client disconnected'));
});

server.listen(8080, () => {
  console.log('HTTPS running at https://localhost:8080');
});
```

4. Phone Motion Sender (public/phone.html)

This page runs on the iPhone and streams accelerometer data to the Node server over WebSockets. It uses the DeviceMotion API + permission request + continuous sending.

```
<!DOCTYPE html>
<html>
<body style="background:#111; color:white; font-family:monospace;">
<h2>iPhone Motion Streaming</h2>
<button id="startBtn">Start motion streaming</button>

<script>
let socket;

document.getElementById('startBtn').onclick = async () => {
  const perm = await DeviceMotionEvent.requestPermission();
  if (perm !== 'granted') {
    alert('Motion permission denied');
    return;
  }
  startWS();
};

function startWS() {
  socket = new WebSocket('wss://YOUR_IP:8080');
  socket.onopen = () => console.log('[PHONE] WS connected');

  window.addEventListener('devicemotion', ev => {
    if (!socket || socket.readyState !== 1) return;

    const x = ev.accelerationIncludingGravity.x || 0;
    const y = ev.accelerationIncludingGravity.y || 0;
    const z = ev.accelerationIncludingGravity.z || 0;

    socket.send(JSON.stringify({
      type: 'motion',
      x, y, z,
      t: Date.now()
    }));
  });
}
</script>
</body>
</html>
```

5. p5.js Viewer (public/sketch.js)

The laptop sketch:
• Polls `/motion` for accelerometer data
• Connects via Web Bluetooth to Garmin HR
• Uses HR + motion to animate a circle

```
let heartRate = 0;
let motionX = 0;
let motionY = 0;
let motionZ = 0;

// -------- poll motion -----------
async function pollMotion() {
  try {
    const res = await fetch('/motion');
    const data = await res.json();
    motionX = data.x;
    motionY = data.y;
    motionZ = data.z;
  } catch (err) {
    console.error('motion poll error', err);
  }
}

// -------- Garmin HR via Web Bluetooth -----------
let device, server, hrChar;

async function connectHeartRate() {
  try {
    device = await navigator.bluetooth.requestDevice({
      filters: [{ services: ['heart_rate'] }]
    });

    server = await device.gatt.connect();
    const service = await server.getPrimaryService('heart_rate');
    hrChar = await service.getCharacteristic('heart_rate_measurement');

    await hrChar.startNotifications();
    hrChar.addEventListener('characteristicvaluechanged', handleHR);
  } catch (err) {
    console.error('Bluetooth error:', err);
  }
}

function handleHR(ev) {
  const data = ev.target.value;
  const flags = data.getUint8(0);
  const is16 = flags & 0x01;
  heartRate = is16 ? data.getUint16(1, true) : data.getUint8(1);
}

// -------- p5 -----------
function setup() {
  createCanvas(800, 600);
  textFont('monospace');
  textSize(18);

  const btn = createButton('Connect Garmin HR');
  btn.mousePressed(connectHeartRate);

  setInterval(pollMotion, 100);
}

function draw() {
```

```
  background(15);

  fill(255);
  text(`HR: ${heartRate}`, 20, 40);
  text(`motionX: ${motionX.toFixed(2)}`, 20, 70);
  text(`motionY: ${motionY.toFixed(2)}`, 20, 100);
  text(`motionZ: ${motionZ.toFixed(2)}`, 20, 130);

  const baseR = map(heartRate || 80, 40, 120, 30, 300, true);
  const jitter = map(abs(motionX)+abs(motionY)+abs(motionZ), 0, 40, 0, 100, true);

  const cx = width/2 + random(-jitter, jitter);
  const cy = height/2 + random(-jitter, jitter);

  noStroke();
  circle(cx, cy, baseR);
}
```

6. Reflection: Using ChatGPT for Technical Development

ChatGPT contributed to:
• Designing the full motion → server → viewer pipeline
• Building a secure WebSocket-based motion stream
• Debugging iPhone permission issues
• Implementing Web Bluetooth HR reading for Garmin
• Writing working templates for all files
• Helping me iterate until the system was stable

This document is submitted as a full technical record of how ChatGPT assisted the project.