Using ChatGPT to Add +24h Predicted Weather to a Node + p5.js System

Technical Walkthrough with Code (for submission)

1. Original Prompt (asked inside ChatGPT)

“Next task, in a new independent sketch is to source live weather data as an input to a p5 sketch. Even better, a predicted weather source. Lead me through a step-by-step guide of how to update my current node, server, sketch setup to include these weather stats. For now just outputted as numbers in the sketch.”

2. Overview of What Was Built

Using ChatGPT, I extended an existing Node.js + p5.js setup that was already handling:
• Live heart rate data from a Garmin HR strap via Web Bluetooth (in the laptop browser)
• Live motion data (x, y, z accelerometer) streamed from an iPhone via a Node/WebSocket server

The goal was to add a third data stream:
• A +24 hour interpolated local weather forecast, so that at any real time t the sketch shows the predicted conditions at t + 24 hours.

The integration has two sides:
1) Node server: fetch hourly forecast data from Open-Meteo, compute the target time (+24h), interpolate between the surrounding hourly points, and expose the blended values at a new /weather24 endpoint.
2) p5.js viewer: periodically poll /weather24, store the returned values in global variables, and render them as text alongside the existing HR and motion values.

3. Node Server Changes (server.js)

Below is the code (as generated and refined with ChatGPT) that I added to my existing HTTPS Node/Express server. The server already served my p5 sketch and provided /motion for phone accelerometer data. These additions enabled weather forecasting and interpolation.

```
// -------- weather forecast config --------
const LAT = 51.5074;    // London latitude (change if needed)
const LON = -0.1278;    // London longitude

let forecastCache = null;
let forecastFetchedAt = 0;
const FORECAST_TTL_MS = 60 * 60 * 1000; // refresh forecast at most once per hour

function buildForecastUrl() {
  const base = 'https://api.open-meteo.com/v1/forecast';
  const params = new URLSearchParams({
    latitude: LAT,
    longitude: LON,
    hourly: [
      'temperature_2m',
      'precipitation_probability',
      'rain',
      'cloudcover',
      'windspeed_10m'
    ].join(','),
    forecast_days: '3',
    timezone: 'auto'
  });
  return `${base}?${params.toString()}`;
}

async function getForecast() {
  const now = Date.now();
  if (forecastCache && now - forecastFetchedAt < FORECAST_TTL_MS) {
    return forecastCache;
  }

  const url = buildForecastUrl();
  const res = await fetch(url);
  if (!res.ok) {
    throw new Error(`Forecast error: ${res.status}`);
  }
  const data = await res.json();
  forecastCache = data;
  forecastFetchedAt = now;
  return data;
}

// find indices i0, i1 such that time[i0] <= target < time[i1]
function findBracketingHours(timeArray, targetDate) {
  let i0 = -1;
  let i1 = -1;
  for (let i = 0; i < timeArray.length - 1; i++) {
    const t0 = new Date(timeArray[i]);
    const t1 = new Date(timeArray[i + 1]);
    if (t0 <= targetDate && targetDate < t1) {
      i0 = i;
      i1 = i + 1;
      break;
    }
  }
  return { i0, i1 };
}

// -------- +24 hour interpolated weather endpoint --------
```

```
app.get('/weather24', async (req, res) => {
  try {
    const forecast = await getForecast();
    const hourly = forecast.hourly;

    const times = hourly.time;
    const now = new Date();
    const target = new Date(now.getTime() + 24 * 60 * 60 * 1000); // now + 24h

    const { i0, i1 } = findBracketingHours(times, target);
    if (i0 === -1 || i1 === -1) {
      return res.status(500).json({ error: 'No forecast window around +24h' });
    }

    const t0 = new Date(times[i0]);
    const t1 = new Date(times[i1]);
    const alpha = (target - t0) / (t1 - t0); // 0..1 between the two hours

    function lerp(a, b, t) {
      return a + (b - a) * t;
    }

    const temp0  = hourly.temperature_2m[i0];
    const temp1  = hourly.temperature_2m[i1];

    const cloud0 = hourly.cloudcover[i0];
    const cloud1 = hourly.cloudcover[i1];

    const rainProb0 = hourly.precipitation_probability[i0];
    const rainProb1 = hourly.precipitation_probability[i1];

    const rain0  = hourly.rain[i0];
    const rain1  = hourly.rain[i1];

    const wind0  = hourly.windspeed_10m[i0];
    const wind1  = hourly.windspeed_10m[i1];

    const payload = {
      now: now.toISOString(),
      target_time: target.toISOString(),
      from_time: t0.toISOString(),
      to_time: t1.toISOString(),
      alpha,

      temperature_24h: lerp(temp0, temp1, alpha),
      cloudcover_24h: lerp(cloud0, cloud1, alpha),
      precip_prob_24h: lerp(rainProb0, rainProb1, alpha),
      rain_24h: lerp(rain0, rain1, alpha),
      windspeed_24h: lerp(wind0, wind1, alpha)
    };

    res.json(payload);
  } catch (err) {
    console.error('weather24 error', err);
    res.status(500).json({ error: 'weather24 error' });
  }
});
```

4. p5.js Viewer Changes (sketch.js)

The p5 sketch already handled drawing a circle based on heart rate and motion, using values fetched from /motion and from the Web Bluetooth HR characteristic. With ChatGPT's help, I added global weather variables, a polling function, and text output to verify everything was working.

```
// -------- weather +24h state --------
let temp24 = null;
let cloud24 = null;
let rainProb24 = null;
let rain24 = null;
let wind24 = null;
let alpha24 = null;
let targetTime24 = '';

// -------- poll +24h weather from server --------
async function pollWeather24() {
  try {
    const res = await fetch('/weather24');
    const data = await res.json();

    temp24 = data.temperature_24h;
    cloud24 = data.cloudcover_24h;
    rainProb24 = data.precip_prob_24h;
    rain24 = data.rain_24h;
    wind24 = data.windspeed_24h;
    alpha24 = data.alpha;
    targetTime24 = data.target_time;

    console.log('[WEATHER24]', data);
  } catch (err) {
    console.error('[WEATHER24] poll error', err);
  }
}

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

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

  // poll motion 10x per second (already in place)
  setInterval(pollMotion, 100);

  // poll +24h weather every 30 seconds
  pollWeather24(); // initial call
  setInterval(pollWeather24, 30 * 1000);
}

function draw() {
  background(15);

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

  // weather numbers (+24h forecast)
  const ty = temp24 != null ? temp24.toFixed(1) : '--';
  const cy = cloud24 != null ? cloud24.toFixed(0) : '--';
  const py = rainProb24 != null ? rainProb24.toFixed(0) : '--';
  const wy = wind24 != null ? wind24.toFixed(1) : '--';
```

```
  text(`Temp +24h: ${ty} °C`, 20, 170);
  text(`Cloud +24h: ${cy} %`, 20, 200);
  text(`Rain prob +24h: ${py} %`, 20, 230);
  text(`Wind +24h: ${wy} km/h`, 20, 260);

  if (targetTime24) {
    textSize(12);
    text(`Target time: ${targetTime24}`, 20, 290);
    textSize(18);
  }

  // existing visual code (e.g. circle driven by HR + motion)
  const baseR = map(heartRate || 80, 40, 100, 30, 300, true);
  const jitter = map(motionMagnitude, 10, 30, 0, 100, true);

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

  noStroke();
  circle(cx, cy, baseR * 2);
}
```

5. Reflection on Using ChatGPT

Throughout this process I used ChatGPT as a collaborative coding partner. I described my existing setup (Node server, WebSocket motion stream, Web Bluetooth heart rate, p5.js viewer) and my conceptual goal: to incorporate predictive weather data as a slow, evolving environmental layer.

ChatGPT helped by:
• Selecting an appropriate free weather API (Open-Meteo)
• Explaining why forecast data is low-frequency and how interpolation can keep the animation alive
• Designing the +24 hour sliding-window concept
• Writing the Node.js helper functions and /weather24 route
• Writing the p5.js polling function and text output scaffolding
• Guiding testing and verification (e.g. using console logs and checking the target time and alpha values)

The final system is now ready to use these weather values not only as text, but as parameters controlling colour, density, motion, and other expressive aspects of the visual. This document is submitted as evidence of how AI assistance was used in the technical development of the work.