

🕒

convert arduino code to p5js



Converting Arduino code to p5.js is not a direct, line-by-line translation. The process depends on whether your Arduino sketch reads sensor data to be visualized or controls physical outputs, but the general workflow involves two separate programs that communicate via serial. [🔗](#)

The key difference

- **Arduino code (C/C++)** runs on a microcontroller to read physical inputs and control electronic outputs. It lacks a graphical interface and cannot interact with a web browser.
- **p5.js code (JavaScript)** runs in a web browser and is used for creating visual graphics, animations, and interactive experiences. It cannot directly control an Arduino's pins. [🔗](#)

Workflow for converting and connecting your code

The most common approach is to use the p5.js Web Serial library to facilitate communication between your p5.js sketch in a browser and your Arduino board connected via USB. [🔗](#)

Step 1: Write the Arduino code

Keep your Arduino sketch simple. Its main job is to read sensor data or button presses and send that information as a string over the serial port. [🔗](#)

Example Arduino code: Read a potentiometer

This sketch reads the value of a potentiometer on pin A0 and sends it over serial, followed by a newline character.

cpp

```
const int potPin = A0;
int sensorValue = 0;

void setup() {
  Serial.begin(9600); // Initialize serial communication
}

void loop() {
  sensorValue = analogRead(potPin); // Read the analog value
  Serial.println(sensorValue);      // Print the value with a newline
  delay(50);                       // Delay to prevent flooding the serial port
}
```

Use code with caution.



Step 2: Set up your p5.js sketch

You'll need an HTML file to include the p5.js and p5.webserial libraries, along with your sketch.js file. [🔗](#)

index.html

html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <script src="https://cdn.jsdelivr.net/npm/p5@1.9.4/lib/p5.js"></script>
  <script src="https://unpkg.com/p5-webserial@0.1.1/build/p5.webserial.js"></script>
  <script src="sketch.js"></script>
</head>
<body>
  <main></main>
</body>
</html>
```

Ask anything



📺📺📺 14 sites

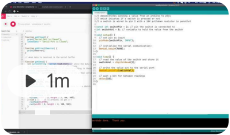


p5.js and Arduino serial communication - Send a...

Mar 28, 2020 — between p5 and the microcontroller. bu...



YouTube · Scott Fitzg...

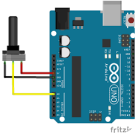


How to send data to p5.js from Arduino

How to send data to p5... * How to send data to p5. js from Arduino. * ...



Creative Technology Lab Wiki



Tutorial: Serial Communication with Arduino and p5.js - Medium

Nov 6, 2018 — js to listen to Arduino. Open the code package from...



Medium



Show all

javascript

```
let serial; // Variable for the p5.webserial library
let inData = 0; // Variable to hold incoming serial data
let connectButton;

function setup() {
  createCanvas(600, 400);

  // Initialize the p5.webserial library
  serial = new p5.WebSerial();

  // Create a button to connect to the serial port
  connectButton = createButton('Connect');
  connectButton.position(10, 10);
  connectButton.mousePressed(connectPort);

  // Set up event listeners for serial communication
  serial.on('data', serialEvent); // When new data comes in, call serialEvent
  serial.on('error', serialError); // If there's an error, call serialError
}

function draw() {
  background(220);

  // Map the incoming sensor data (0-1023) to a canvas size (e.g., 0-width)
  let circleSize = map(inData, 0, 1023, 50, 300);

  // Draw a circle whose size is controlled by the sensor data
  fill(0, 100, 255);
  noStroke();
  ellipse(width / 2, height / 2, circleSize, circleSize);

  // Display the current sensor value
  fill(0);
  textSize(24);
  textAlign(CENTER, CENTER);
  text('Sensor Value: ' + inData, width / 2, height - 50);
}

function connectPort() {
  serial.requestPort(); // Request a port selection from the user
}

function serialEvent() {
  // Read the incoming serial data
  let dataString = serial.readLine();
  if (dataString) {
    // Trim whitespace and convert the string to a number
    inData = Number(dataString.trim());
  }
}

function serialError(err) {
  console.log("Something went wrong with the serial port: " + err);
}
```

Use code with caution.



Explanation of the p5.js sketch

- `setup()` : This function initializes the canvas, creates an instance of the `p5.WebSerial` library, and creates the "Connect" button. It also registers event-handling functions for incoming serial data and errors.
- `draw()` : This function runs continuously. It clears the background, then uses the `map()` function to convert the `inData` value from the Arduino's analog range (`0-1023`) to a suitable pixel size for the p5.js canvas.
- `connectPort()` : This function is called when the button is pressed, prompting the user to select the correct serial port for their Arduino.
- `serialEvent()` : This callback function is triggered whenever new data arrives from the Arduino. It reads the incoming string, cleans up any whitespace (`trim()`), converts it

To create a menu with a 2x3 button matrix in p5.js, you will define a `Button` class to handle each button's properties, use nested loops to create a 2D array of buttons, and write logic to detect and respond to clicks on each button. [↗](#)

This approach avoids creating individual variables for every button and provides a more scalable and organized structure for your menu. [↗](#)

Step 1: Create a `Button` class

The `Button` class will manage the position, size, text, and click behavior for each button. [↗](#)

javascript

```
class Button {
  constructor(x, y, w, h, label) {
    this.x = x;
    this.y = y;
    this.w = w;
    this.h = h;
    this.label = label;
    this.isClicked = false;
  }

  // Check if the mouse is inside the button's boundaries
  isMouseOver() {
    return mouseX > this.x && mouseX < this.x + this.w &&
      mouseY > this.y && mouseY < this.y + this.h;
  }

  // Draw the button
  display() {
    if (this.isMouseOver()) {
      fill(150); // Highlight color when mouse is over
    } else {
      fill(200);
    }
    rect(this.x, this.y, this.w, this.h);
    fill(0);
    textAlign(CENTER, CENTER);
    text(this.label, this.x + this.w / 2, this.y + this.h / 2);
  }
}
```

Use code with caution.



Step 2: Set up the p5.js sketch with a button matrix

The `setup()` function creates a 2D array to hold the button objects. The `draw()` function will then display these buttons. [↗](#)

javascript

```
let buttonMatrix = [];
const rows = 2;
const cols = 3;
const buttonWidth = 100;
const buttonHeight = 50;
const xMargin = 50;
const yMargin = 50;
const xSpacing = 20;
const ySpacing = 20;

function setup() {
  createCanvas(450, 200);

  // Use nested loops to create the 2x3 button matrix
  for (let r = 0; r < rows; r++) {
    buttonMatrix[r] = []; // Create a new row
    for (let c = 0; c < cols; c++) {
      let x = xMargin + c * (buttonWidth + xSpacing);
      let y = yMargin + r * (buttonHeight + ySpacing);
      let label = `Button ${r * cols + c + 1}`;
    }
  }
}
```

is pressed. Declaring the function mousePressed()...

✖ p5.js ⋮

Responding to Inputs - p5.js

Prerequisites * Step 1 – Create and style your buttons. Start with a...

✖ p5.js ⋮



Button Array - p5.js Web Editor

let button = []; 2. 3. createCanvas(540, 540); } 8. background(0); 10. for (let i = 0; i < 8; i++) { for (var...

✖ p5.js Web Editor ⋮

Show all

```
        buttonMatrix[r][c].display();
    }
}
}
```

Use code with caution.

Step 3: Add click event logic

Use the built-in `mousePressed()` function to check for clicks on each button in the matrix and then perform a specific action. [🔗](#)

```
javascript

function mousePressed() {
  for (let r = 0; r < rows; r++) {
    for (let c = 0; c < cols; c++) {
      let button = buttonMatrix[r][c];
      if (button.isMouseOver()) {
        // Log the button that was clicked and perform a specific action
        console.log(`${button.label} was clicked!`);

        // Example: change a global state variable
        // (You would have to define and use `selectedButton` elsewhere)
        // selectedButton = button;

        // Example: trigger a function based on the button clicked
        // if (button.label === "Button 1") {
        //   doSomething();
        // }
      }
    }
  }
}
```

Use code with caution.

Putting it all together

For a full, runnable sketch, combine the three code snippets above into a single `sketch.js` file and include it in your `index.html` file. [🔗](#)

`sketch.js` (Complete)

```
javascript

let buttonMatrix = [];
const rows = 2;
const cols = 3;
const buttonWidth = 100;
const buttonHeight = 50;
const xMargin = 50;
const yMargin = 50;
const xSpacing = 20;
const ySpacing = 20;

class Button {
  constructor(x, y, w, h, label) {
    this.x = x;
    this.y = y;
    this.w = w;
    this.h = h;
    this.label = label;
  }

  isMouseOver() {
    return mouseX > this.x && mouseX < this.x + this.w &&
      mouseY > this.y && mouseY < this.y + this.h;
  }

  display() {
    if (this.isMouseOver()) {
      fill(150);
    }
  }
}
```

```
createCanvas(450, 200);
for (let r = 0; r < rows; r++) {
  buttonMatrix[r] = [];
  for (let c = 0; c < cols; c++) {
    let x = xMargin + c * (buttonWidth + xSpacing);
    let y = yMargin + r * (buttonHeight + ySpacing);
    let label = `Button ${r * cols + c + 1}`;
    buttonMatrix[r][c] = new Button(x, y, buttonWidth, buttonHeight, label);
  }
}

function draw() {
  background(220);
  for (let r = 0; r < rows; r++) {
    for (let c = 0; c < cols; c++) {
      buttonMatrix[r][c].display();
    }
  }
}

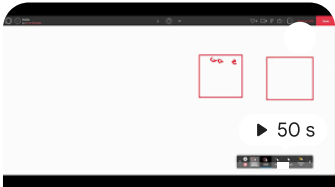
function mousePressed() {
  for (let r = 0; r < rows; r++) {
    for (let c = 0; c < cols; c++) {
      let button = buttonMatrix[r][c];
      if (button.isMouseOver()) {
        console.log(`${button.label} was clicked!`);
        // Add your custom logic here
      }
    }
  }
}
```

Use code with caution.



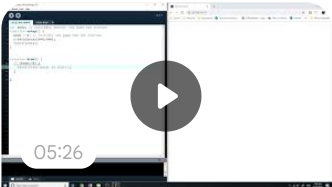
Designing Interactive User Interfaces: Creating Men...

Tech Head Online
YouTube



Open Processing - P5js - Menu System

Christopher Ockerby
YouTube



p5.js How to display text menu before game starts

Magic Monk
YouTube

See more

AI responses may include mistakes. [Learn more](#)



add serial communications for Arduino and p5js

Adding serial communication to the p5.js button matrix example requires an Arduino sketch to send data and updates to the p5.js code to read and act on that data. A common pattern is to have the Arduino send a single character indicating which button was pressed. [🔗](#)

Step 1: Arduino code

This sketch will send a unique character over the serial port for each button press. You will need to build the 2x3 button matrix on a breadboard and connect it to your Arduino. [🔗](#)

Hardware:

- 6 Pushbuttons: For your 2x3 matrix.
- Arduino board: Such as an Uno or Nano.
- Resistors: Six pull-down resistors (e.g., 10kΩ).
- Jumper wires [🔗](#)

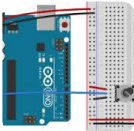
22 sites



Lab: Serial Input to p5.js Using the p5.webserial Library - ITP

To communicate with your microcontroller serially, you're going...

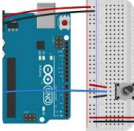
NYU



Lab: Serial Input to P5.js Using the p5.serialport library - ITP

Install the p5. Serialcontrol App. Download the latest version of the P...

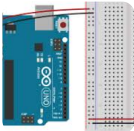
NYU



Lab: Serial Output From p5.js Using the p5.webserial Library

Overview. When you use the p5. webserial library for P5. js, it uses th...

NYU



Show all



```
// Define the digital pins connected to each button
const int buttonPins[6] = {2, 3, 4, 5, 6, 7};
const char buttonChars[6] = {'1', '2', '3', '4', '5', '6'};
const int debounceDelay = 50; // Debounce delay in milliseconds

int lastButtonState[6] = {HIGH, HIGH, HIGH, HIGH, HIGH, HIGH};
long lastDebounceTime[6] = {0, 0, 0, 0, 0, 0};

void setup() {
  Serial.begin(9600); // Set baud rate
  for (int i = 0; i < 6; i++) {
    pinMode(buttonPins[i], INPUT);
  }
}

void loop() {
  for (int i = 0; i < 6; i++) {
    int reading = digitalRead(buttonPins[i]);

    if (reading != lastButtonState[i]) {
      lastDebounceTime[i] = millis();
    }

    if ((millis() - lastDebounceTime[i]) > debounceDelay) {
      if (reading != lastButtonState[i]) {
        lastButtonState[i] = reading;
        if (lastButtonState[i] == HIGH) {
          Serial.println(buttonChars[i]);
        }
      }
    }
  }
}
```

Use code with caution.



Step 2: p5.js code updates

This will build upon the previous 2x3 button matrix example. You'll add the p5.webserial library to handle communication and update the sketch to highlight a button based on serial input. [↗](#)

1. Update index.html

Add the p5.webserial library to your HTML file to enable serial communication from the browser. [↗](#)

html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <script src="https://cdn.jsdelivr.net/npm/p5@1.9.4/lib/p5.js"></script>
  <script src="https://unpkg.com/p5-webserial@0.1.1/build/p5.webserial.js"></script>
  <script src="sketch.js"></script>
</head>
<body>
  <main></main>
</body>
</html>
```

Use code with caution.



2. Update sketch.js

The p5.js sketch will need new variables for the serial connection and a way to track the active button. The `setup()` function is modified to include a "Connect" button that opens the serial port chooser, and the `draw()` function will update the button's appearance based on incoming serial data. [↗](#)

javascript

```
let buttonMatrix = [];
const rows = 2;
```

```
let activeLabel = ''; // Tracks the label of the currently active button

class Button {
  constructor(x, y, w, h, label) {
    this.x = x;
    this.y = y;
    this.w = w;
    this.h = h;
    this.label = label;
  }

  isMouseOver() {
    return mouseX > this.x && mouseX < this.x + this.w &&
      mouseY > this.y && mouseY < this.y + this.h;
  }

  display() {
    // Check if the button is active based on serial data or mouse hover
    if (this.label === `Button ${activeLabel}` || this.isMouseOver()) {
      fill(150);
    } else {
      fill(200);
    }
    rect(this.x, this.y, this.w, this.h);
    fill(0);
    textAlign(CENTER, CENTER);
    text(this.label, this.x + this.w / 2, this.y + this.h / 2);
  }
}

function setup() {
  createCanvas(450, 200);
  serial = new p5.WebSerial(); // Create a new instance of p5.WebSerial

  // Create the serial connection button
  connectButton = createButton('Connect');
  connectButton.position(10, 10);
  connectButton.mousePressed(connectPort);

  // Set up event listeners for serial communication
  serial.on('data', serialEvent);

  for (let r = 0; r < rows; r++) {
    buttonMatrix[r] = [];
    for (let c = 0; c < cols; c++) {
      let x = xMargin + c * (buttonWidth + xSpacing);
      let y = yMargin + r * (buttonHeight + ySpacing);
      let label = `Button ${r * cols + c + 1}`;
      buttonMatrix[r][c] = new Button(x, y, buttonWidth, buttonHeight, label);
    }
  }
}

function draw() {
  background(220);
  for (let r = 0; r < rows; r++) {
    for (let c = 0; c < cols; c++) {
      buttonMatrix[r][c].display();
    }
  }
}

function connectPort() {
  serial.requestPort(); // Show the serial port selector
}

function serialEvent() {
  let data = serial.readLine(); // Read the serial string
  if (data) {
    let trimmedData = data.trim();
    if (trimmedData) {
      activeLabel = trimmedData; // Set the active label to the incoming data
      console.log("Received from Arduino:", activeLabel);
    }
  }
}
```



```
}  
}
```

Use code with caution.

How to use

- 1. Prepare your Arduino:** Upload the `arduino_button_matrix.ino` sketch to your board.
- 2. Open in a compatible browser:** Open the `index.html` file in a compatible browser (like Chrome or Edge) that supports the Web Serial API.
- 3. Connect via serial:** Click the "Connect" button in your p5.js sketch. A prompt will appear asking you to select your Arduino's serial port.
- 4. Test the button matrix:** Press the physical buttons on your breadboard. The corresponding digital buttons on your p5.js sketch will be highlighted. [↗](#)

AI responses may include mistakes. [Learn more](#)

🔗👍💬

preview in p5js

The best way to preview your p5.js project is by **running a local web server**. A web browser cannot load files for security reasons when you just double-click the `index.html` file. A web server resolves this issue and allows the serial communication to work correctly. [↗](#)

Here are three common methods for running a local server to preview your p5.js project.

Method 1: Use the VS Code Live Server extension

If you use Visual Studio Code, this is the most convenient method. The extension launches a local development server for you automatically. [↗](#)

- 1. Install the extension:** Open Visual Studio Code and go to the Extensions view (the icon on the left that looks like four squares). Search for **"Live Server"** by Ritwick Dey and click **Install**.

2. Start the server:

1. Right-click your `index.html` file and select **Open with Live Server**.
2. Alternatively, click the **"Go Live"** button in the bottom-right corner of the VS Code window.

- 3. Preview in browser:** Your default browser will open automatically and display your project. Any changes you save will cause the browser to reload automatically. [↗](#)

Method 2: Use the Python `http.server` module

If you have Python installed, you can start a simple web server from your terminal or command prompt with one line of code. [↗](#)

- 1. Open a terminal:** Navigate to your project folder where the `index.html` file is located.

1. On Windows, use Command Prompt or PowerShell.
2. On macOS or Linux, use Terminal.

2. Run the command:

1. For **Python 3**: Type `python3 -m http.server 8000` and press Enter.
2. For **Python 2 (older systems)**: Use `python -m SimpleHTTPServer 8000`.

- 3. Open in browser:** Open your web browser and go to `http://localhost:8000`. [↗](#)

Method 3: Use the p5.js Web Editor

The official p5.js online editor is a complete, self-contained environment for writing and previewing p5.js sketches, including those that use serial communication. [↗](#)

- 1. Go to the editor:** Navigate to `https://editor.p5js.org/` in your browser.

🔍

14 sites

⋮

How to Use Live Server in Visual Studio Code for Real-Time ...

Nov 16, 2024 — How to Use Live Server in Visual Studio Code for Real-Time Web Development.

📖 Medium ⋮

1.1 Web Editor and the Console - p5.js Tutorial

Aug 30, 2020 — in this tutorial. we are going to ge...

📺 YouTube · xin xin ⋮

Simple HTTP Server in Python

Dec 26, 2021 — and the first one is quite simple so we'r...

📺 YouTube · NeuralNine ⋮

Show all



How to create an on-screen button in the P5.js...

Jason Erdreich
YouTube



8.5: Interacting with the DOM using Sliders, Butto...

The Coding Train
YouTube



9.3: More about Transformations - p5.js...

The Coding Train
YouTube

See more

AI responses may include mistakes. [Learn more](#)

