mac hc 04 bluetooth

An HC-04 is a Bluetooth serial communication module used in projects like Arduino, not a consumer accessory for Macs. **A Mac cannot connect to an HC-04 module directly using the standard Bluetooth settings because the HC-04 only functions as a "slave," meaning it can only accept incoming connections, not initiate them**. It is designed to be a serial-to-Bluetooth bridge for microcontrollers, not a standard peripheral.

HC-04 vs. HC-05

Many users confuse the HC-04 with the more flexible HC-05 model, which can act as both a "master" and a "slave." The key difference is the role they can perform.

- **HC-04 (Slave only):** Can only be detected and connected to by other Bluetooth devices.
- **HC-05 (Master/Slave):** Can both detect other devices and accept incoming connections.

How to use an HC-04 with a Mac

To use an HC-04 with a Mac, you must first connect it to a microcontroller, such as an Arduino. This setup turns the module into a wireless serial interface for your project.

Hardware requirements

- A Mac running macOS
- An HC-04 Bluetooth module
- A microcontroller (like an Arduino)
- Jumper wires
- A breadboard (optional)

Step 1: Wire the HC-04 to a microcontroller

Connect the HC-04 to your Arduino following these standard connections:

- `VCC` to the Arduino's 5V pin
- `GND` to the Arduino's GND pin
- `TXD` to the Arduino's `RX` pin
- `RXD` to the Arduino's `TX` pin (may require a voltage divider to protect the 3.3V HC-04)

Step 2: Set up the Arduino sketch

Upload a sketch to the Arduino that uses the serial communication to pass data back and forth through the HC-04. A common way is to echo any received data back to the Mac.

17 sites

HC-05 and HC-06 zs-040 Bluetooth modules. First Look
Oct 4, 2014 — First Look. ... Update: If you have modules that have a blue...
Martyn Currey

HC-04 Bluetooth-compatible 5.0 Module HC04 Dual-mode ...
Item description Report Item. ... The HC-04 is a Bluetooth serial...
AliExpress

What are the fundemental differences Bluetooth ...
Dec 17, 2014 — 2 Answers. Sorted by: 9. simplicity is concerned, HC05 would be
Arduino Stack Exchange

Show all

Ask anything

2. Find the HC-04 in the list of discovered devices and click **Connect**.
3. If prompted, enter the default pairing code, which is usually `1234`.

Step 4: Use a serial terminal on your Mac

Once paired, your Mac will create a virtual serial port for the HC-04 connection.

1. Use a serial terminal program (e.g., CoolTerm, the Arduino IDE's Serial Monitor, or `screen` in the Terminal).
2. The serial port will have a name like `/dev/cu.HC-04-DevB` or something similar. You can find this port in **System Information > Hardware > Bluetooth**.
3. Set the correct baud rate in your terminal program (e.g., 9600, the default for many modules).
4. You can now send and receive data wirelessly between your Mac and the Arduino project via the HC-04.

example with arduino uno code and wiring

An HC-04 Bluetooth module can be connected to an Arduino Uno using the `SoftwareSerial` library to enable wireless communication with another Bluetooth device.

This example sets up the Arduino to:

- Listen for data sent from a connected Bluetooth device (like a Mac or smartphone).
- Read that data and print it to the Arduino's built-in serial monitor.
- Receive text from the built-in serial monitor and send it back over Bluetooth.

This configuration acts as a serial-to-Bluetooth pass-through, allowing you to use your Mac's serial terminal to interact with the Arduino wirelessly.

Hardware requirements

- Arduino Uno
- HC-04 Bluetooth Module
- Breadboard
- Jumper wires
- 1x 1kΩ resistor
- 1x 2kΩ resistor (or two 1kΩ resistors in series)

Wiring diagram

32 sites

Things I learned about HC-05 and Arduino
Jul 2, 2021 — I was confused but many posts not being clear about...
Arduino Forum

configuring HC-04 Bluetooth modem arduino!!!!
Jan 24, 2013 — */ #include SoftwareSe mySerial(2,3); // RX, TX. void setup() { //
Arduino Forum

Lesson 36: Get Started with Bluetooth Module
In this project, we demonstrate how to communicate with a Bluetooth...
SunFounder

Show all

- **RXD:** Receive data.

Connections

The HC-04 module operates at a 3.3V logic level, but the Uno operates at 5V. To prevent damage to the module, you must use a voltage divider on the Arduino's TX pin (which sends 5V data) to the HC-04's RX pin. The HC-04's TX pin can be connected directly to the Arduino's RX pin because the 3.3V signal is sufficient for the Uno to read.

Wiring table:

Arduino Uno Pin	HC-04 Pin	Connection Details
5V	VCC	Connect directly to power the module.
GND	GND	Connect directly.
Pin 3	TXD	Connect directly. This is the SoftwareSerial RX line on the Arduino.
Pin 2	RXD	Connect via a voltage divider.

Voltage divider connection:

1. Connect the Arduino's **Pin 2** to one end of the **2kΩ resistor**.
2. Connect the other end of the **2kΩ resistor** to one end of the **1kΩ resistor**.
3. Connect the other end of the **1kΩ resistor** to the Arduino's **GND**.
4. Connect the **HC-04's RXD pin** to the point between the two resistors.

Arduino Uno code

This sketch uses `SoftwareSerial` to create a virtual serial port on digital pins 2 and 3, which is separate from the default hardware serial port (pins 0 and 1). This is important because the hardware port is used for communication with your computer via the USB cable.

cpp

```cpp
#include <SoftwareSerial.h>

// Initialize the SoftwareSerial port on pins 2 and 3
// RX on pin 3, TX on pin 2
SoftwareSerial BTSerial(3, 2);

void setup() {
  // Start the hardware serial port (for communicating with the computer)
  Serial.begin(9600);
  Serial.println("Arduino with HC-04 Bluetooth module ready.");
  Serial.println("Pair with the HC-04 using the default PIN (1234).");
  Serial.println("Send a message from your computer's Serial Monitor to send over B

  // Start the software serial port (for communicating with the Bluetooth module)
  // The default baud rate for most HC-0x modules is 9600
  BTSerial.begin(9600);
}
```

```
  // Read data coming from the computer's serial monitor and send it to Bluetooth
  if (Serial.available()) {
    String serialData = Serial.readString();
    BTSerial.print(serialData);
  }
}
```

Use code with caution.

How to use with a Mac

1. **Upload the code:** Connect your Arduino Uno to your Mac with a USB cable and upload the sketch.
2. **Pair with Bluetooth:**
   1. Open **System Settings** > **Bluetooth** on your Mac.
   2. The HC-04 should appear as a discovered device (it may have a generic name like `HC-04`). Click **Connect**.
   3. Enter the default PIN, which is usually `1234`.
3. **Find the serial port:**
   1. After pairing, a new serial port will be available.
   2. You can find the name in **System Information** > **Hardware** > **Bluetooth**. Look for the HC-04 device and note its `Outgoing` serial port name. It will be similar to `/dev/tty.HC-04-DevB`.
4. **Connect with a terminal:**
   1. Open your preferred terminal program (like CoolTerm) or use the command-line `screen` utility.
   2. Using `screen`, you would enter: `screen /dev/tty.HC-04-DevB 9600`
   3. Anything you type into the terminal will now be sent to the Arduino via Bluetooth, and any data the Arduino sends will appear in your terminal.

AI responses may include mistakes. Learn more

example as a remote control for an r/c toy

An HC-04 can be used with an Arduino to build a simple Bluetooth-controlled r/c toy like a car. The Arduino receives commands from your computer or smartphone via the HC-04 and uses a motor driver to control the movement of DC motors.

This example outlines the hardware, wiring, and code required to build a simple two-wheel-drive car controlled by a computer's serial monitor or a Bluetooth terminal app on

23 sites

How to make Bluetooth controlled Car using...

Aug 28, 2022 — hi guys welcome back to 23rd part...

YouTube · Tech at Ho...

How to Make a Bluetooth Controlled Simplistic RC Car

Dec 11, 2017 — We will need 2 motors and an H-bridge to drive the motor. ...

Instructables

- L298N Motor Driver Module (or similar H-Bridge driver)
- 2x DC gear motors with wheels
- A chassis or mounting plate for the components
- Breadboard
- 9V battery and connector for the Arduino (or other power source)
- External battery pack for the motor driver (e.g., 4x AA batteries or a Li-ion pack)
- Jumper wires
- 1x 1kΩ resistor and 1x 2kΩ resistor for the voltage divider

Wiring diagram

HC-04 connections

As detailed in the previous example, the HC-04 connects to the Arduino's `SoftwareSerial` pins:

- **HC-04 VCC** to **Arduino 5V**
- **HC-04 GND** to **Arduino GND**
- **HC-04 TXD** to **Arduino Pin 3**
- **HC-04 RXD** to **Arduino Pin 2** (via voltage divider)

L298N Motor Driver connections

- **L298N** `12V` to your external motor battery's positive terminal.
- **L298N** `GND` to the external battery's negative terminal and **Arduino GND**.
- **L298N** `OUT1` and `OUT2` to the terminals of your first motor.
- **L298N** `OUT3` and `OUT4` to the terminals of your second motor.
- **L298N** `IN1` to **Arduino Pin 5**
- **L298N** `IN2` to **Arduino Pin 6**
- **L298N** `IN3` to **Arduino Pin 7**
- **L298N** `IN4` to **Arduino Pin 8**

Powering the system

- The Arduino is powered by its 9V battery or a USB connection.
- The motor driver and motors must be powered by a separate, higher-capacity battery pack.

Arduino Uno code

This code reads single-character commands sent over Bluetooth and uses them to

Arduino Project Hub

Show all

```
// L298N motor driver control pins
const int motor1Pin1 = 5;
const int motor1Pin2 = 6;
const int motor2Pin1 = 7;
const int motor2Pin2 = 8;

void setup() {
  // Set motor control pins as outputs
  pinMode(motor1Pin1, OUTPUT);
  pinMode(motor1Pin2, OUTPUT);
  pinMode(motor2Pin1, OUTPUT);
  pinMode(motor2Pin2, OUTPUT);

  // Stop all motors initially
  stopMotors();

  // Start the hardware serial for debugging
  Serial.begin(9600);
  Serial.println("RC toy ready. Awaiting commands over Bluetooth.");

  // Start the software serial for Bluetooth
  BTSerial.begin(9600);
}

void loop() {
  if (BTSerial.available()) {
    char command = BTSerial.read();

    switch (command) {
      case 'F': // Forward
        moveForward();
        break;
      case 'B': // Backward
        moveBackward();
        break;
      case 'L': // Turn left
        turnLeft();
        break;
      case 'R': // Turn right
        turnRight();
        break;
      case 'S': // Stop
        stopMotors();
        break;
    }
  }
}

void moveForward() {
  digitalWrite(motor1Pin1, HIGH);
  digitalWrite(motor1Pin2, LOW);
  digitalWrite(motor2Pin1, HIGH);
  digitalWrite(motor2Pin2, LOW);
}

void moveBackward() {
  digitalWrite(motor1Pin1, LOW);
```

```
  digitalWrite(motor2Pin1, HIGH);
  digitalWrite(motor2Pin2, LOW);
}

void turnRight() {
  digitalWrite(motor1Pin1, HIGH);
  digitalWrite(motor1Pin2, LOW);
  digitalWrite(motor2Pin1, LOW);
  digitalWrite(motor2Pin2, HIGH);
}

void stopMotors() {
  digitalWrite(motor1Pin1, LOW);
  digitalWrite(motor1Pin2, LOW);
  digitalWrite(motor2Pin1, LOW);
  digitalWrite(motor2Pin2, LOW);
}
```

Use code with caution.

Controlling the toy

After assembling the hardware and uploading the code, you can control the toy from your Mac or smartphone.

From a Mac

1. **Pair** with the HC-04 in **System Settings > Bluetooth**.
2. Find the outgoing serial port name in **System Information > Hardware > Bluetooth**.
3. Open a terminal and connect using the `screen` command: `screen /dev/tty.HC-04-DevB 9600`.
4. Type the control commands (**F, B, L, R, S**) and press Enter to send them.

From an Android phone

1. Install a **Bluetooth terminal app** from the Google Play Store (e.g., "Bluetooth Serial Terminal" or "Arduino Bluetooth Controller").
2. **Pair** your phone with the HC-04 module.
3. Open the app and connect to the HC-04 device.
4. Use the app's buttons or keyboard to send the command characters to control the car.

Remote Control Car Using Arduino for Beginners — Lee Curiosity

Build a Bluetooth RC Car with Arduino | Low-Cost... — UtGo Tech

Using an RC Controller with an Arduino [With code an... — Andy Vickers

more youtube videos

YouTube provides numerous video tutorials for creating Bluetooth-controlled RC cars using an Arduino and an HC-series Bluetooth module. While many tutorials feature the more flexible HC-05, the same principles apply to the HC-04, with the main difference being the HC-04's "slave-only" functionality.

PROJECT: Tinkering RC Cars Adding HC-06...
TechToTinker
YouTube

Build A Bluetooth RC Interface / Scamp2 / HC-...
0033mer
YouTube

Remote Control Car Using Arduino for Beginners
Lee Curiosity
YouTube

See more

Basic tutorials and introductions

- **"Diy : Control Your Rc Car With Bluetooth!"** by Electro Guru is a comprehensive, step-by-step guide. It explains how to build a car controlled by a Bluetooth app on your smartphone, covering:
  - Connecting the HC-04 module to an Arduino.
  - Using an L298N motor driver to control the DC motors.
  - The Arduino sketch that interprets commands from the Bluetooth module.
  - A link to the complete code and circuit diagram is included in the video description.
- **"How To Make Smartphone Controlled Car At Home"** by The Electronics is a popular video demonstrating a build for a Bluetooth-controlled rock crawler. The video details the full construction process, from the chassis to the electronics, and includes connections for the HC-05 (or HC-04), Arduino, and L298N motor driver.

Advanced and alternative approaches

- **"Build a Bluetooth RC Interface"** by The Scamp Life demonstrates building a more advanced interface to control a robot or car. While it uses a different microcontroller, the video offers valuable insights into creating a more feature-rich Bluetooth interface, including:
  - Implementing forward, reverse, and turning commands.
  - Adding controls for headlights, a horn, and other custom functions.
- **"Bluetooth Controlled Car with Automatic Braking"** on Hackster.io provides a tutorial for an RC car with obstacle avoidance. The project uses a HC-SR04 ultrasonic sensor in addition to an HC-05 Bluetooth module. The video details how to use a motor shield instead of a breadboard for more compact construction.

12 sites

Diy : Control Your Rc Car With Bluetooth! - YouTube
Apr 12, 2024 — "Welcome to our Arduino Bluetooth RC car tutorial! In...
YouTube

Build A Bluetooth RC Interface / Scamp2 / HC-0...
Jan 1, 2022 — okay in this video. we are going to look...
YouTube · 0033mer

How to make a Bluetooth Controlled RC Car [Codes...
Dec 30, 2017 — okay so you all know that we have...
YouTube · Creative cr...

Show all

convert this to a midi musical instrument that uses bluetooth to send values to a computer.

To convert your Arduino and HC-04 RC car project into a MIDI musical instrument, you must replace the car's components with musical inputs and reconfigure the software to send MIDI-formatted messages over Bluetooth. For the most straightforward experience with a Mac, you will need a serial-to-MIDI application to convert the HC-04's output.

Hardware changes

The L298N motor driver and DC motors from the RC car project are no longer needed. Instead, you will use sensors or buttons to create musical inputs. The HC-04 module will remain for wireless communication.

Required components

- **Arduino Uno**
- **HC-04 Bluetooth module**
- **Inputs:**
  - Pushbuttons for drum pads or note triggers.
  - Potentiometers for continuous controller (CC) messages (e.g., volume, panning).
  - Other sensors like an HC-SR04 ultrasonic sensor for a theremin-like effect.
- **Voltage divider parts:**
  - 1x 1kΩ resistor
  - 1x 2kΩ resistor

Wiring

Connect the HC-04 to the Arduino just as you did for the RC car, including the voltage divider for the RX pin. Add your new musical inputs to the remaining digital and analog pins.

Input wiring examples:

- **Pushbuttons:** Connect one leg to a digital pin and the other to ground. Use the Arduino's `INPUT_PULLUP` setting to avoid external resistors.
- **Potentiometers:** Connect the outer legs to 5V and GND. Connect the middle leg to an analog pin (e.g., A0, A1, etc.) to read the changing resistance.

Software setup

The RC car's code needs to be replaced with a sketch that:

20 sites

MIDI for the Arduino - Build a MIDI Output Circuit

Mar 18, 2015 — finally we'll connect a wire from pin fiv...

YouTube · Notes and ...

Set up Bluetooth MIDI devices in Audi on Mac

Make your Mac a Bluetooth peripheral MIDI Setup app on your Mac, choose W

Apple Support

Send and Receive MIDI With Arduino : 11 Steps (with Pictures) -...

Send and Receive MIDI With Arduino * Step 1: Bytes and Bits. To understan...

Instructables

Show all

Code to send MIDI over Bluetooth:

cpp

```cpp
#include <SoftwareSerial.h>
#include <MIDI.h>

// SoftwareSerial pins for HC-04
SoftwareSerial BTSerial(3, 2);

// Create a new instance of the MIDI_Interface
MIDI_CREATE_INSTANCE(SoftwareSerial, BTSerial, midi);

// Pin for a button input
const int buttonPin = 4;
bool lastButtonState = HIGH;

// Pin for a potentiometer input
const int potPin = A0;
int lastPotValue = -1;

void setup() {
  // Start the hardware serial for debugging
  Serial.begin(9600);
  Serial.println("Arduino MIDI controller ready.");

  // Start the Bluetooth serial port
  BTSerial.begin(9600);

  // Set up input pins
  pinMode(buttonPin, INPUT_PULLUP);
}

void loop() {
  // --- BUTTON INPUT ---
  // Read the button state
  bool currentButtonState = digitalRead(buttonPin);

  // Check for a note-on event (button press)
  if (currentButtonState == LOW && lastButtonState == HIGH) {
    midi.sendNoteOn(60, 127, 1); // Note C4, max velocity, channel 1
  }
  // Check for a note-off event (button release)
  else if (currentButtonState == HIGH && lastButtonState == LOW) {
    midi.sendNoteOff(60, 0, 1); // Note C4, velocity 0, channel 1
  }
  lastButtonState = currentButtonState;

  // --- POTENTIOMETER INPUT ---
  // Read the analog value
  int potValue = analogRead(potPin);

  // Map the analog value (0-1023) to a MIDI CC value (0-127)
  int midiCCValue = map(potValue, 0, 1023, 0, 127);

  // Send the MIDI Continuous Controller message if the value has changed
  if (abs(midiCCValue - lastPotValue) > 1) { // Add a filter to reduce data
```

Mac software setup

The Mac will not automatically recognize the HC-04 as a MIDI device. You need a program to convert the serial data from the HC-04 into virtual MIDI messages that your music software can understand.

1. **Install a serial-to-MIDI bridge:**
    1. **Hairless MIDI:** This is a free, cross-platform tool that reads serial data and converts it to MIDI.
    2. **LoopMIDI:** This is a virtual MIDI port for Windows, but a similar setup is needed on macOS using the built-in "Audio MIDI Setup" utility.
2. **Configure the MIDI setup on your Mac:**
    1. Open `Applications > Utilities > Audio MIDI Setup`.
    2. Choose **Window > Show MIDI Studio**.
    3. Click the **Bluetooth icon** and connect to your HC-04 module.
    4. This may create a virtual MIDI port for the HC-04, which you can then use in your DAW.
    5. Alternatively, use the serial-to-MIDI bridge application you installed to manage the connection.
3. **Connect to a Digital Audio Workstation (DAW):**
    1. Open a DAW like **GarageBand, Ableton Live, or Logic Pro**.
    2. In the program's settings, find the MIDI/Control Surface section.
    3. The virtual MIDI port created by the HC-04 (or the bridging software) should appear as an available MIDI input.
    4. Select the correct input, and you can now use your custom Arduino instrument to trigger notes and control parameters.

instead of using the analog input for volume, change it to pitch bend.

You can modify the Arduino sketch to change the potentiometer input from sending MIDI volume (Control Change #7) to pitch bend messages. The Arduino `MIDI` library provides a dedicated function `midi.sendPitchBend()` for this purpose.

The MIDI pitch bend message uses a 14-bit value, which offers much higher resolution than the 7-bit value of a standard control change. The range for pitch bend is from 0 to 16,383, with the center (no pitch bend) at 8,192.

16 sites

Pitch Bend - Brendan Vavra
3. Updating the code to map joystick input to MIDI pitch bend. The...
GitHub

MIDI Messages | MIDI for Makers - Adafruit Learning System
Mar 2, 2022 — A MIDI CC message contains the CC number and the...
Adafruit Learning System

Send midi pitch bend with a joystick - Arduino Forum

Show all

```
// SoftwareSerial pins for HC-04
SoftwareSerial BTSerial(3, 2);

// Create a new instance of the MIDI_Interface
MIDI_CREATE_INSTANCE(SoftwareSerial, BTSerial, midi);

// Pin for a button input
const int buttonPin = 4;
bool lastButtonState = HIGH;

// Pin for a potentiometer input for pitch bend
const int pitchBendPotPin = A0;
int lastPitchBendValue = -1;

void setup() {
  // Start the hardware serial for debugging
  Serial.begin(9600);
  Serial.println("Arduino MIDI controller ready. Potentiometer controls pitch bend.

  // Start the Bluetooth serial port
  BTSerial.begin(9600);

  // Set up input pins
  pinMode(buttonPin, INPUT_PULLUP);
}

void loop() {
  // --- BUTTON INPUT ---
  // Read the button state
  bool currentButtonState = digitalRead(buttonPin);

  // Check for a note-on event (button press)
  if (currentButtonState == LOW && lastButtonState == HIGH) {
    midi.sendNoteOn(60, 127, 1); // Note C4, max velocity, channel 1
  }
  // Check for a note-off event (button release)
  else if (currentButtonState == HIGH && lastButtonState == LOW) {
    midi.sendNoteOff(60, 0, 1); // Note C4, velocity 0, channel 1
  }
  lastButtonState = currentButtonState;

  // --- PITCH BEND POTENTIOMETER INPUT ---
  // Read the analog value
  int rawAnalogValue = analogRead(pitchBendPotPin);

  // Map the analog value (0-1023) to a MIDI Pitch Bend value (0-16383)
  int midiPitchBendValue = map(rawAnalogValue, 0, 1023, 0, 16383);

  // Add a simple threshold to reduce sending pitch bend messages for tiny changes
  if (abs(midiPitchBendValue - lastPitchBendValue) > 10) {
    midi.sendPitchBend(midiPitchBendValue, 1);
    lastPitchBendValue = midiPitchBendValue;
  }
}
```

- Changed the `map()` range from `0, 127` to `0, 16383`. This converts the potentiometer's 10-bit value to the 14-bit MIDI pitch bend range.
- **Added a new threshold of** `10` for detecting changes in the potentiometer. This is a higher value than before because the overall range is larger, and it prevents sending too many messages due to analog noise.

After updating the code

1. **Upload the new sketch** to your Arduino Uno.
2. **Verify your connections** for the potentiometer and HC-04 are unchanged.
3. **Connect** your Mac to the HC-04 over Bluetooth, as you did before.
4. **Use Hairless MIDI** or your DAW's MIDI settings to route the incoming data.
5. In your DAW, assign the pitch bend input to an instrument track.
6. Now, when you trigger a note with the button, you can use the potentiometer to bend its pitch up or down.

AI responses may include mistakes. Learn more

add 10 more buttons and map them to notes, assume whole notes, c major scale, 4th octave.

To add 10 buttons mapped to the C Major scale in the 4th octave, you can use a combination of digital input pins and a matrix arrangement to conserve Arduino pins. This configuration allows you to send MIDI `NoteOn` and `NoteOff` messages for each button press.

Hardware changes

Use a keypad matrix

An Arduino Uno has a limited number of digital pins. A keypad matrix is an efficient way to read multiple buttons using fewer pins. A 4x4 matrix, for example, requires only eight pins to read 16 buttons. You can build a custom matrix with 10 buttons or use a pre-built 4x4 keypad.

Required components

- **Arduino Uno**
- **HC-04 Bluetooth module** (with voltage divider for RX)
- **10x Pushbuttons**
- **10x 1kΩ resistors** (for matrix wiring)
- 1x 1kΩ resistor and 1x 2kΩ resistor (for HC-04 voltage divider)

5 sites

Arduino Button Matrix 4x4: A Comprehensive Guide
Apr 13, 2024 — If you're looking for a way to control multiple buttons with...
www.niceone-keypad.com

Would a switch matrix work with many pressed?
Jan 22, 2022 — If the hardware has only matrix, the hardware won't allow readin
Electrical Engineering Stack Excha...

MIDI_Class Class Reference - Arduino
Call the begin method in the setup() fu Arduino. void. sendNoteOn (byte NoteN
SourceForge

Show all

option and requires only seven digital pins. If you use a custom setup, adjust the pin connections accordingly.

Button Matrix Wiring:

- **Rows (Arduino outputs):** `Pin 5`, `Pin 6`, `Pin 7`
- **Columns (Arduino inputs):** `Pin 8`, `Pin 9`, `Pin 10`, `Pin 11`

Button-to-Pin mapping:
The connections for a standard 3x4 keypad can be found online. The following is a general guide to connecting buttons in a matrix:

1. Connect the row pins to the rows of your button grid.
2. Connect the column pins to the columns of your button grid.
3. Add diodes on each button connection to prevent "ghosting" if multiple buttons are pressed simultaneously.

Software setup

The updated Arduino sketch will include the standard HC-04 and MIDI library setup, but it will also use the `Keypad` library to handle reading the button matrix.

MIDI note numbers for C Major (4th octave)

- C4: 60
- D4: 62
- E4: 64
- F4: 65
- G4: 67
- A4: 69
- B4: 71
- C5: 72
- D5: 74
- E5: 76

Arduino code

cpp

```
#include <SoftwareSerial.h>
#include <MIDI.h>
#include <Keypad.h>

// SoftwareSerial pins for HC-04
SoftwareSerial BTSerial(3, 2);
```

```
  {'9','*','0','#'} // '*' and '#' buttons can be used for other functions
};
byte rowPins[ROWS] = {5, 6, 7}; // Connect to rows
byte colPins[COLS] = {8, 9, 10, 11}; // Connect to columns

// MIDI note numbers for the C Major scale (4th octave)
int noteNumbers[ROWS][COLS] = {
  {60, 62, 64, 65}, // C4, D4, E4, F4
  {67, 69, 71, 72}, // G4, A4, B4, C5
  {74, 76, 0, 0}    // D5, E5 (remaining notes can be assigned later)
};

// Create the Keypad object
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

void setup() {
  // Start the hardware serial for debugging
  Serial.begin(9600);
  Serial.println("Arduino MIDI keyboard ready.");

  // Start the Bluetooth serial port
  BTSerial.begin(9600);
}

void loop() {
  // Check for button presses
  char key = keypad.getKey();

  if (key != NO_KEY) {
    // Determine which key was pressed
    int rowIndex = -1;
    int colIndex = -1;

    // Find the pressed key's position in the matrix
    for (int r = 0; r < ROWS; r++) {
      for (int c = 0; c < COLS; c++) {
        if (keys[r][c] == key) {
          rowIndex = r;
          colIndex = c;
          break;
        }
      }
      if (rowIndex != -1) break;
    }

    if (rowIndex != -1 && colIndex != -1) {
      int note = noteNumbers[rowIndex][colIndex];
      if (note != 0) {
        midi.sendNoteOn(note, 127, 1); // NoteOn message
      }
    }
  } else {
    // This simple approach sends a NoteOff for all keys when no key is pressed.
    // A more advanced approach would track individual button states.
    for (int r = 0; r < ROWS; r++) {
      for (int c = 0; c < COLS; c++) {
        int note = noteNumbers[r][c];
        if (note != 0) {
```

Improvements

- **Track button states:** A more robust approach would be to track the `NoteOn` and `NoteOff` states for each button individually. This allows for polyphony (playing multiple notes at once) and more reliable performance.
- **Debouncing:** The Keypad library often includes internal debouncing, but for very fast performance, more advanced techniques might be needed.
- **Dedicated MIDI libraries:** For more complex MIDI controllers, consider using a dedicated MIDI library for Arduino, which can handle more complex MIDI messages.

convert this code to p5js

Directly converting Arduino C++ code to p5.js is not possible because they are different languages running on different platforms. The Arduino runs on a microcontroller, while p5.js runs in a web browser using JavaScript. The communication layer must be re-engineered, as the HC-04 sends serial data to the computer, not the browser.

To achieve the same functionality in p5.js, you must follow these steps:

1. **Refactor the Arduino sketch** to send raw sensor data as text strings over Bluetooth serial instead of formatted MIDI messages.
2. **Use a serial-to-web bridge** to make the Bluetooth serial data accessible to the browser. You cannot directly access the Bluetooth device from the browser.
3. **Write a p5.js sketch** to read the serial data, parse the sensor values, and use the p5.WebMidi library to send MIDI messages to your computer's music software.

Step 1: Updated Arduino sketch

This sketch reads the keypad and pitch bend potentiometer, then prints the raw values to the serial port. The button states are represented by `NoteOn` and `NoteOff` messages, and the pitch bend by a continuous value.

cpp

```cpp
#include <SoftwareSerial.h>
#include <Keypad.h>

// SoftwareSerial pins for HC-04
SoftwareSerial BTSerial(3, 2);

// Button Matrix setup
const byte ROWS = 3;
const byte COLS = 4;
char keys[ROWS][COLS] = {
```

2 sites

Lab: Serial Input to P5.js Using the p5.serialport library - ITP
js Serialport Library. To communicate with your microcontroller serially,...
NYU

Serial over Bluetooth on the web | Blog - Chrome for Developers
Oct 13, 2023 — The Web Bluetooth API and the Web Serial API allow we...
Chrome for Developers

```
const int pitchBendPotPin = A0;
int lastPitchBendValue = -1;

void setup() {
  Serial.begin(9600);
  BTSerial.begin(9600);
  Serial.println("Arduino ready for P5.js. Sending raw data.");
}

void loop() {
  // Check for button presses
  char key = keypad.getKey();
  if (key != NO_KEY) {
    BTSerial.print("key:");
    BTSerial.println(key);
  }

  // Read pitch bend pot and send changes
  int rawAnalogValue = analogRead(pitchBendPotPin);
  int mappedPitchBendValue = map(rawAnalogValue, 0, 1023, 0, 16383);

  if (abs(mappedPitchBendValue - lastPitchBendValue) > 10) {
    BTSerial.print("pitch:");
    BTSerial.println(mappedPitchBendValue);
    lastPitchBendValue = mappedPitchBendValue;
  }

  delay(10);
}
```

Use code with caution.

Step 2: Bridge serial to the web

You cannot directly access Bluetooth serial from p5.js. You must use a separate application that runs locally on your machine to forward the serial data to the browser.

- **For macOS,** use **Hairless MIDI**. You will have to change your Arduino code to a simple text-based protocol and have Hairless route it to your DAW, as the browser can't get raw serial directly. You can try the **Web Serial API** in Chrome, but it requires HTTPS and a button click for permission.
- **For all platforms,** use `p5.serialserver`. This is a small application that runs on your computer and creates a web socket server to relay data between your serial port and your browser.

Step 3: p5.js sketch using `p5.serialserver` and `WebMidi.js`

This p5.js sketch uses `p5.serialserver` to get data from the Arduino and `WebMidi.js` to send MIDI messages.

`index.html`

```
html
```

```
  </head>
  <body>
    <main></main>
  </body>
</html>
```

Use code with caution.

sketch.js

javascript

```
let serial;
let portName = '/dev/cu.BTSerial-HC-04'; // CHANGE THIS to your HC-04's serial port
let midiOutput;

// Map characters from the keypad to MIDI note numbers
const keyToNoteMap = {
  '1': 60, '2': 62, '3': 64, '4': 65,
  '5': 67, '6': 69, '7': 71, '8': 72,
  '9': 74, '*': 76, '0': 0, '#': 0
};

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

  // Set up p5.serialport
  serial = new p5.SerialPort();
  serial.on('list', gotList);
  serial.on('connected', serverConnected);
  serial.on('open', portOpen);
  serial.on('data', serialEvent);
  serial.on('error', serialError);
  serial.on('close', portClose);
  serial.list();
  serial.open(portName);

  // Set up WebMidi.js
  WebMidi.enable(function (err) {
    if (err) {
      console.log("WebMidi could not be enabled.", err);
    } else {
      console.log("WebMidi enabled!");
      // Display available MIDI outputs in the console
      console.log("Available MIDI Output ports:", WebMidi.outputs);
      // Change this to the name of your MIDI output port
      midiOutput = WebMidi.outputs[0];
    }
  });
}

function draw() {
  background(220);
  text("Listening for MIDI messages from Arduino...", 10, 20);
}
```

```
  if (inString.length > 0) {
    console.log("Received:", inString);
    const parts = inString.split(':');
    if (parts.length === 2) {
      const type = parts[0];
      const value = parts[1].trim();

      if (type === 'key') {
        // Send NoteOn for button press
        const note = keyToNoteMap[value];
        if (note) {
          midiOutput.playNote(note, 1, 100);
        }
      } else if (type === 'pitch') {
        // Send Pitch Bend
        const pitchBendValue = parseInt(value, 10);
        midiOutput.sendPitchBend(pitchBendValue, 1);
      }
    }
  }
}
```

Use code with caution.

How to run the p5.js sketch

1. **Install** `p5.serialserver`: Follow the instructions on the library's GitHub page.
2. **Start** `p5.serialserver`: Run the application on your computer.
3. **Find your port:** When `p5.serialserver` is running, it will list your serial ports. Find the one for your HC-04.
4. **Update** `portName`: Edit the `portName` variable in your p5.js `sketch.js` file with the correct serial port name.
5. **Open** `index.html`: Open the `index.html` file in your browser.
6. **Find MIDI output:** In the p5.js console, find the list of available MIDI output ports and set `midiOutput` to the correct one (e.g., your DAW or a virtual MIDI port).
7. **Map MIDI:** Configure your DAW to listen for MIDI input from the selected port. You can now use your Arduino instrument to trigger notes and control pitch bend.

AI responses may include mistakes. Learn more

make the p5js sketch a p5js app

To package your p5.js sketch as a standalone desktop application, you can use Electron. Electron is a framework that bundles web technologies like HTML, CSS, and JavaScript (including your p5.js sketch) into a cross-platform desktop app.

4 sites

P5.JS with Electron JS— Making Generative Art Distributable ...

Nov 19, 2021 — Settings. QualityAuto. SpeedNormal. Debug log. Video...

Medium

download them from the official Node.js website.

2. **Arduino Code**: The Arduino sketch from the previous step, which prints raw data over the serial port, is still needed. The HC-04 module must also be paired via Bluetooth with your computer.

Step 1: Set up the Electron project

First, create a new directory for your project and use npm to initialize it.

1. **Create a folder** for your project and navigate into it from your terminal:

```sh
mkdir p5-midi-app
cd p5-midi-app
```

Use code with caution.

2. **Initialize the project** with npm:

```sh
npm init -y
```

Use code with caution.

3. **Install Electron** and the `serialport` library:

```sh
npm install electron@latest serialport --save-dev
```

Use code with caution.

Step 2: Create the project files

Create the following files in your project directory.

`package.json`

Update the `package.json` file to include a start script for Electron.

```json
{
  "name": "p5-midi-app",
  "version": "1.0.0",
  "description": "A p5.js MIDI instrument app.",
  "main": "main.js",
  "scripts": {
```

Electron

A set of p5.js templates that run as Electron apps. - GitHub

Electron p5. js Templates. A set of templates to create p5. js Electron...

GitHub

Show all

`main.js`

This file is the main process for your Electron app. It creates the app window and handles the serial port communication from the HC-04, forwarding the data to the p5.js sketch.

The `main.js` file uses Electron modules to create a browser window and load your HTML file. It also sets up serial communication using the `serialport` library, connecting to a specified port and baud rate. A `ReadlineParser` is used to process data received from the serial port. When data is received, it's sent to the renderer process (your p5.js sketch) via Electron's `ipcMain` module. Remember to change the `portName` variable to match your HC-04's serial port. You can find the full script in the referenced web document.

`preload.js`

This script runs in a sandboxed environment before the renderer process loads. It uses `contextBridge` to expose a limited API to the renderer process, allowing the p5.js sketch to safely receive serial data from `main.js` using `ipcRenderer`. The `receiveSerialData` function is exposed, which takes a callback function to handle incoming data. You can find the full script in the referenced web document.

`index.html`

This is a basic HTML file that loads the necessary library files (`p5.js` and `webmidi.min.js`) and your sketch file (`sketch.js`). It also includes a reference to `preload.js` in the `webPreferences` of the `BrowserWindow` in `main.js`. Some basic styling is included to remove default margins and make the canvas a block element. You can find the full HTML in the referenced web document.

`sketch.js`

Your p5.js sketch code is modified to work within the Electron environment. The `setup()` function now initializes WebMidi.js and uses the `window.api.receiveSerialData()` function from `preload.js` to listen for data sent from the main process. The serial data is parsed, and if it corresponds to a key press or pitch bend value, the appropriate MIDI message is sent using WebMidi.js. A mapping of keypad characters to MIDI note numbers is included. The `draw()` function simply sets the background and displays some text. You can find the full sketch code in the referenced web document.

Step 3: Run the app

1. **Download library files:** Download `p5.js` and `WebMidi.min.js` and place them in your project directory.
2. **Start the app:** Run the following command from your project directory:

```sh
npm start
```

Use code with caution.