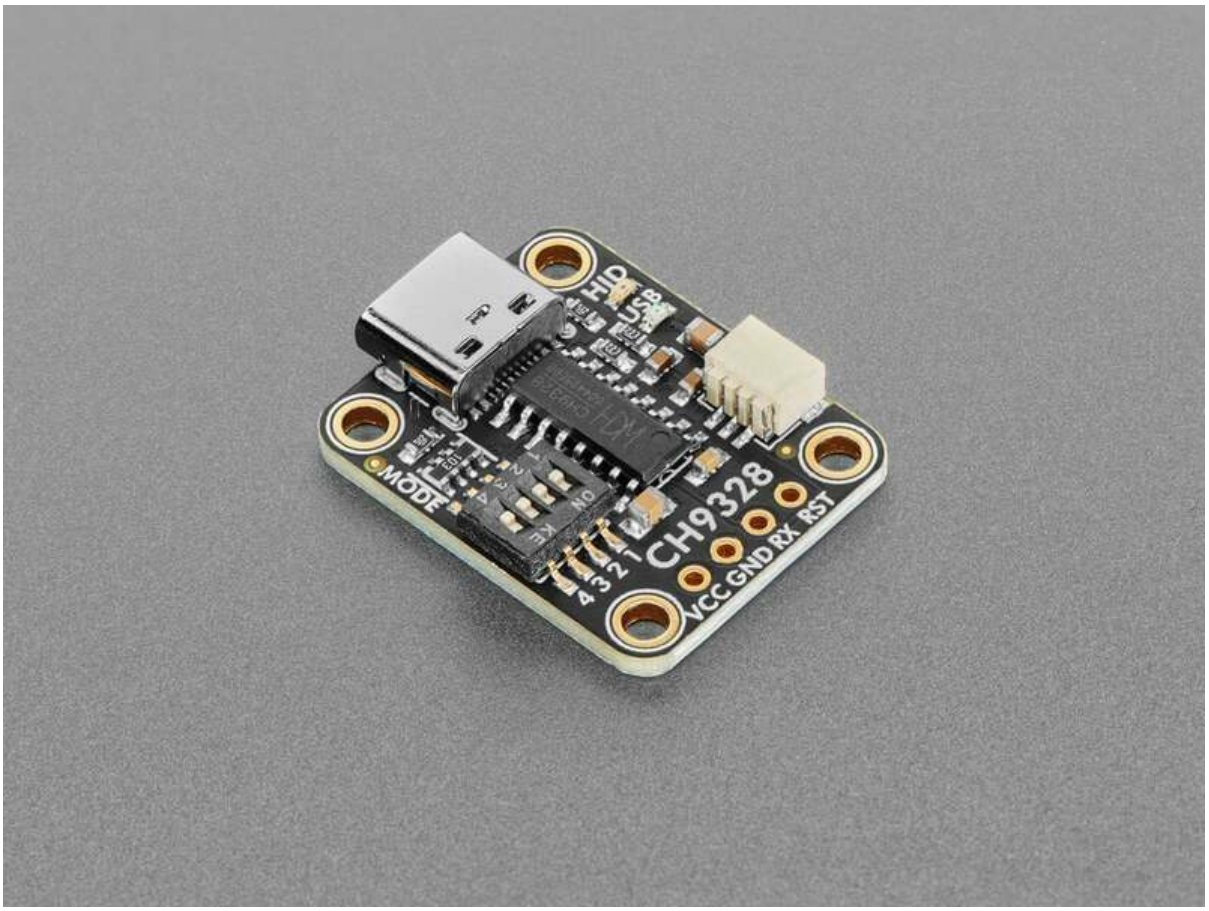
Adafruit CH9328 UART to HID Keyboard Breakout

Created by Liz Clark

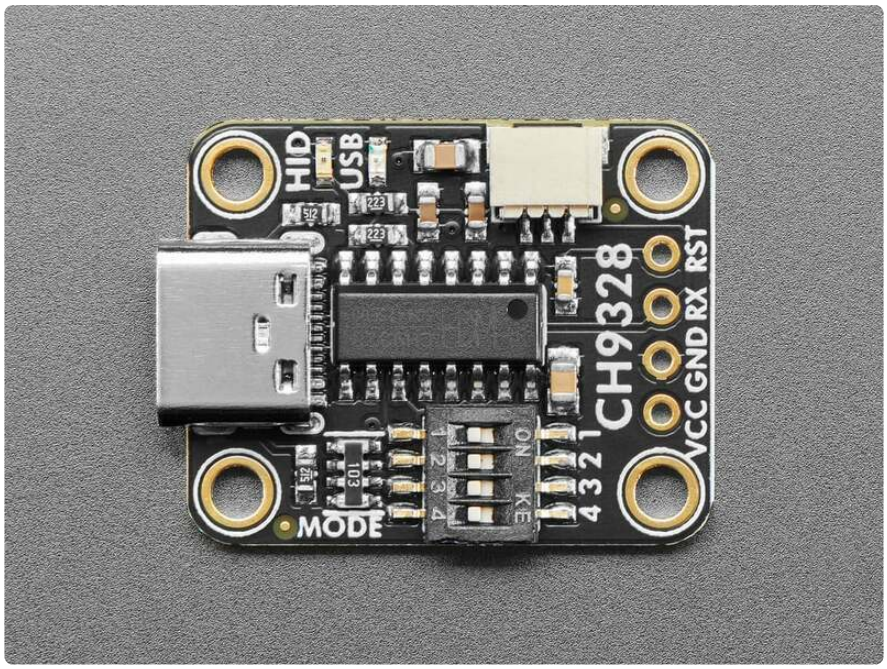


https://learn.adafruit.com/adafruit-ch9328-uart-to-hid-keyboard-breakout

Last updated on 2024-06-24 02:28:26 PM EDT

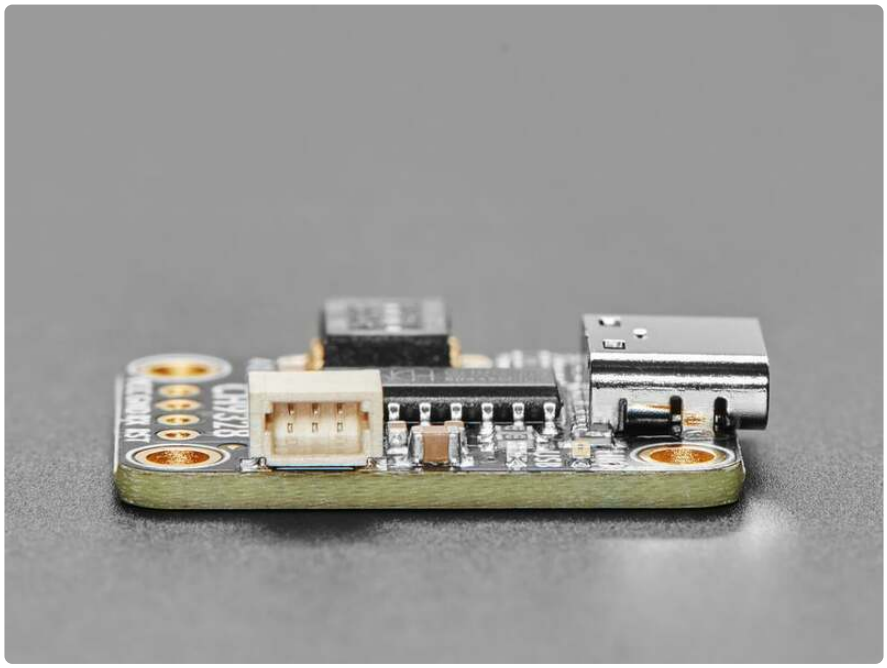
Table of Contents

Overview

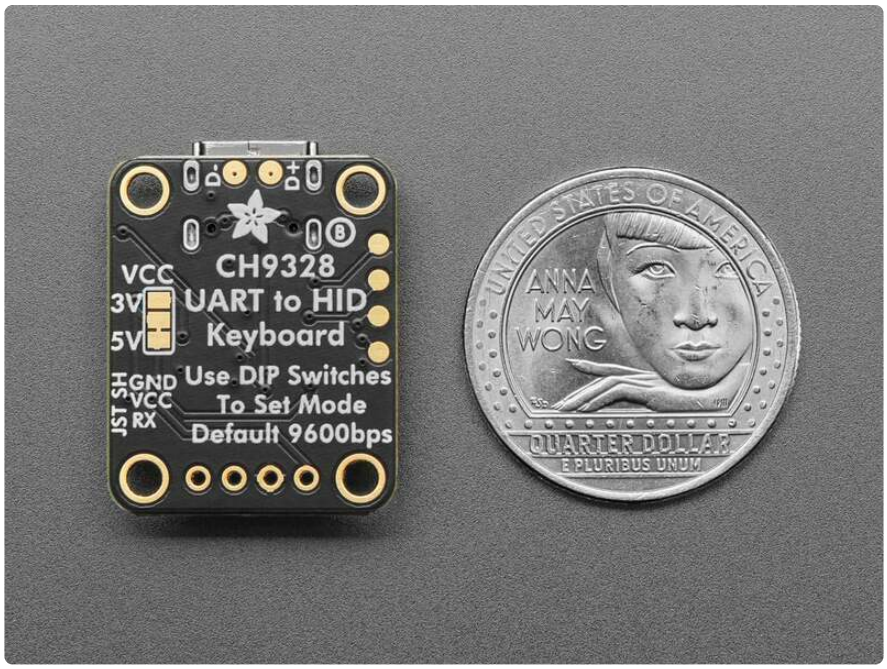


We love using chips with 'native USB' peripherals - that's the magic silicon that lets a microcontroller act like an HID keyboard or mouse or disk drive or MIDI synth. It's a standard addition on SAMD21 (http://adafru.it/4600), RP2040 (https://adafru.it/1a2C), and even the latest ESP32-S2 (https://adafru.it/1a2D) and ESP32-S3 (https://adafru.it/1a2E) boards. But what about when you have a classic ATmega328 Uno? or an original ESP32 or ESP8266? Maybe even a single-board computer like a Raspberry Pi? We would say "sorry...that's not possible", UNTIL NOW!

The CH9328 (https://adafru.it/1a2F) is a funky chip that is basically a programmed microcontroller that enumerates as an every-day HID keyboard and can convert ASCII or 8-byte raw reports, read over a standard serial port UART, into keypresses. So, you can emulate a keyboard even if your chip doesn't have native USB! You do need a hardware or software serial port: some way to generate 9600 baud 3V-logic signal that the CH9328 can read.



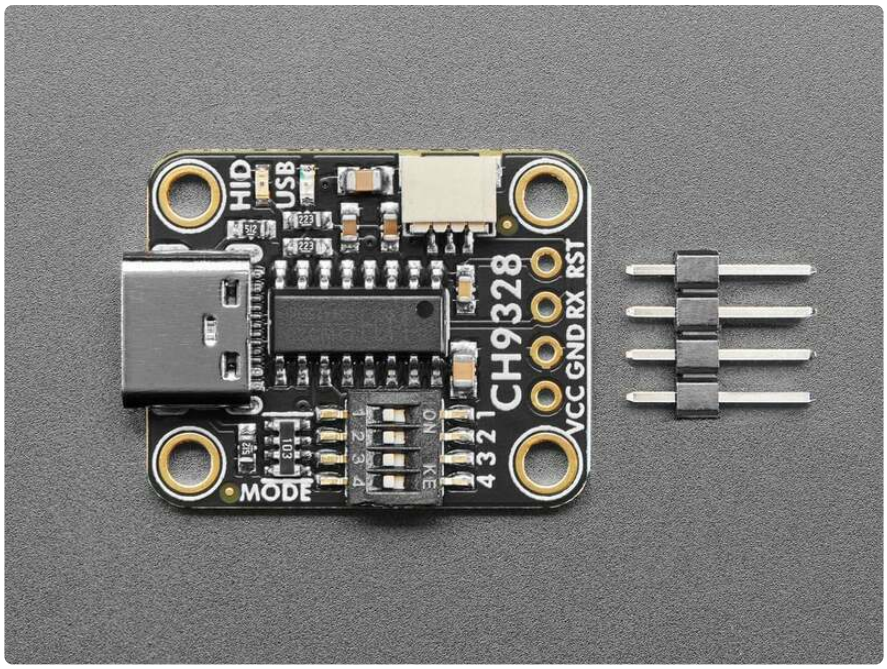
We added all the support circuitry to make this chip easy to use, you may not even need to do any soldering. The CH9328 is connected to a USB Type C port so it's easy to plug into any computer, laptop or even phone/tablet. Then, you can get 5V power from the 5V and Ground pads. There's a UART RX receive input plus a reset line if you want to perform a hard reset. If you want 'solderless' functionality, grab one of our JST SH cables (http://adafru.it/5755): the red line will provide 5V, black is Ground, and the white wire is data in.



You can configure the 'Mode' using the 4 on-board switches, do that before powering it up:

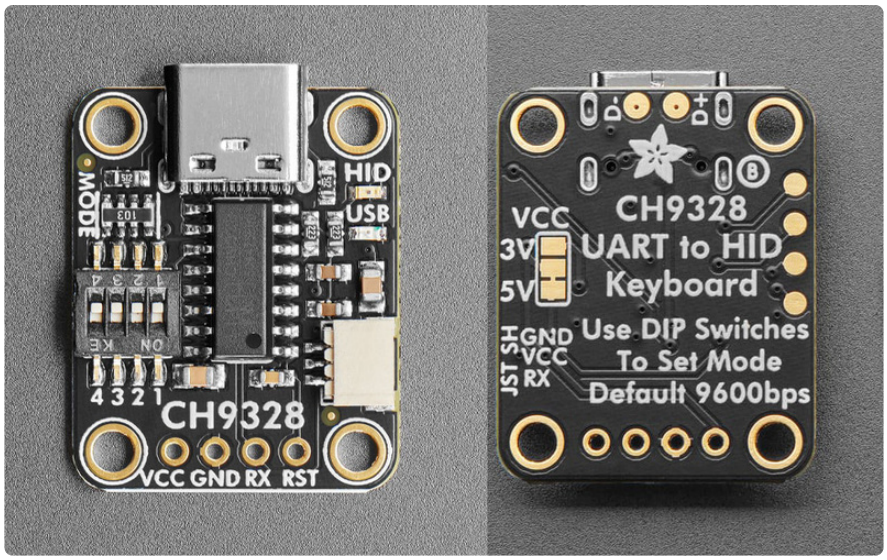
- **Switch #1** is used to configure the "Upload speed" - for most cases it doesn't matter which way it's set
- **If Switch #2, #3 and #4 are all 'ON' the chip is in Mode 0:**
  Visible ASCII characters are sent as-is: if you send ASCII "A", an "A" is typed. If 0x1B is received, it is converted to the Enter key
- **If Switch #2 and #4 are 'ON' and Switch #3 is 'OFF' the chip is in Mode 1:**
  Visible ASCII characters are sent as-is: if you send ASCII "A", an "A" is typed
- **If Switch #2 and #3 are 'ON' and Switch #4 is 'OFF' the chip is in Mode 2:**
  Visible ASCII characters are sent as-is: if you send ASCII "A", an "A" is typed. If 0x28 is received, it is converted to the Enter key
- **If Switch #3 and #4 are 'ON' and Switch #2 is 'OFF' the chip is in Mode 3:**
  This is the mode for sending raw 8-byte HID reports. This is also good if you want more control over the keypresses, or keyholds, or to mimic a non-US keyboard because you can control exactly what control codes are sent and when.

We recommend using mode 3 if possible, but you'll want to use our CH9328 library to do so.



This is a nice and easy way to emulate a keyboard without having to worry about native USB support, gadget mode, or maybe you just want to emulate more than one kind of device. You can also do 'funky' things like have one desktop or single-board computer 'type' into a device such as a computer or mobile device by running our Python code and having it send UART data via a USB-to-UART converter (http://adafru.it/954). Either way, the CH9328 will make it easy to keeb away.

Pinouts



Power

- **USB C Port** - at the top edge of the board is the USB C port. Use this port to connect the breakout to your computer, laptop or even a phone or tablet. This

port provides power and ground to the CH9328, as well as an HID connection to your chosen device.
- **VCC** - this is the power output from the USB C port. It is 5V by default.
- **GND** - common ground for power and logic.

UART

- **RX** - This is the UART **RX** receive input for the CH9328. Connect this pin to the UART TX transmit output pin on your microcontroller.

Reset

- **RST** - This is the **reset** line for the CH9328. If you want to perform a hard reset, set this pin low.

JST SH Port

- **JST SH** (http://adafru.it/5755) - 1mm pitch JST port for use with 3-pin JST SH cables (http://adafru.it/5765). It has connections for:
    - **GND** - common ground for power and data. It is the black wire on the JST SH cable.
    - **VCC** - power output from the USB C port. It is the red wire on the JST SH cable.
    - **RX** - RX input for the CH9328. It is the white wire on the JST SH cable.

Switches

There are four switches on the breakout board labeled **1**, **2**, **3** and **4**. Switch **1** is used to configure the "Upload speed" - when the switch is on, it is set to normal speed and when it is off it is set to high speed (approximately 2x the speed of normal speed). For most cases it doesn't matter which way its set.

Switches **2**, **3** and **4** configure the 'Mode' for the CH9328. You'll need to select your 'Mode' before powering up the board. There are four modes available:

- **Mode 0:**
  Visible ASCII characters are sent as-is: if you send ASCII "A", an "A" is typed. If 0x1B is received, it is converted to the Enter key

- **Mode 1:**
  Visible ASCII characters are sent as-is: if you send ASCII "A", an "A" is typed
- **Mode 2:**
  Visible ASCII characters are sent as-is: if you send ASCII "A", an "A" is typed. If 0x28 is received, it is converted to the Enter key
- **Mode 3:**
  This is the mode for sending raw 8-byte HID reports. This is also good if you want more control over the keypresses, or keyholds, or to mimic a non-US keyboard because you can control exactly what control codes are sent and when.

The modes are selected with these switch combinations:

MODE	1	2	3	4
0	-	ON	ON	ON
1	-	ON	OFF	ON
2	-	ON	ON	OFF
3	-	OFF	ON	ON

VCC Jumper

On the back of the board is the **VCC jumper**. It is outlined in white on the board silk and is labeled **VCC**. If you cut the **5V** pad from the **center pad**, this will disconnect the 5V line output from the VCC pin. If you solder the **3V** pad to the **center pad**, it will connect the 3V output to the VCC pin. This will cause the output voltage from VCC to be 3V.

Status LEDs

There are two LEDs on the front of the board:

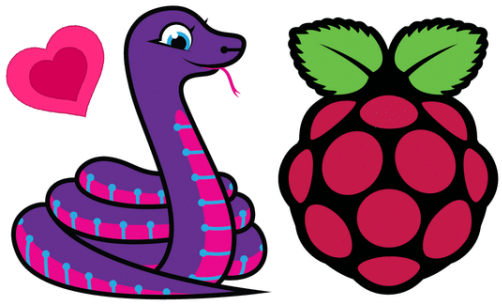
- **HID** - This is the HID USB status LED. It is a yellow LED and it is connected to the ACT pin on the CH9328. When the USB configuration setup is complete, this LED will light up.
- **USB** - This is the serial UART status LED. It is a red LED and it is connected to the T_LED pin on the CH9328. This LED is lit when a serial UART message is

received on the RX pin and is sent via USB. If there is no UART activity, then this LED will be off.

CircuitPython and Python

It's easy to use the **CH9328** with CircuitPython and the Adafruit_CircuitPython_CH9328 (https://adafru.it/1a2G) driver. This driver allows you to easily write Python code to send UART messages to the CH9328.

You can use this driver with any CircuitPython microcontroller board or with a computer that has GPIO, UART and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library (https://adafru.it/BSN). For Blinka, this example was tested with a Raspberry Pi. To use the example as-is with built-in UART, you'll need to follow these configuration steps (https://adafru.it/CEk) outlined in the Blinka Learn Guide.



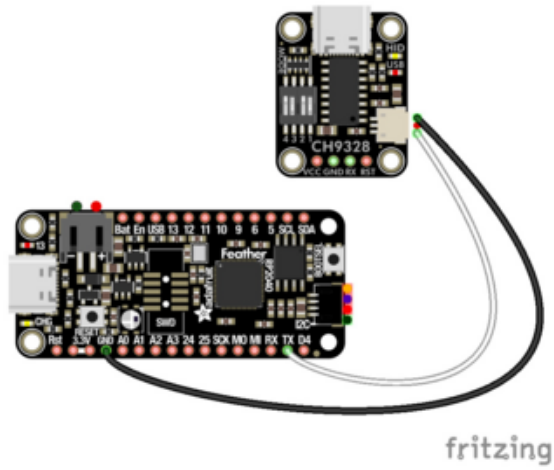
CircuitPython Libraries on Linux and Raspberry Pi
By M. LeBlanc-Williams
UART / Serial

https://learn.adafruit.com/circuitpython-on-raspberrypi-linux/uart-serial

Make sure that the CH9328 onboard switches are set to Mode 3 (switch 2 off, switches 3 and 4 on) before powering it up.

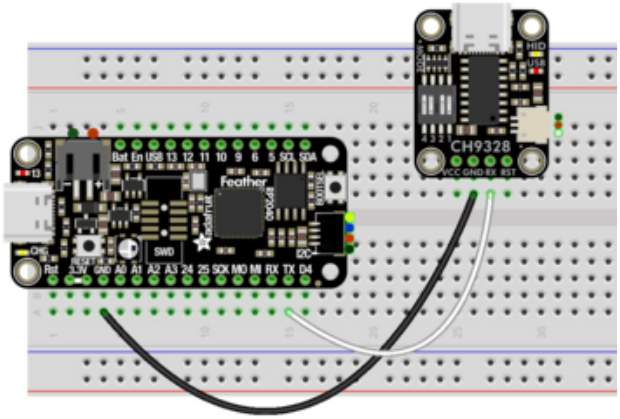
CircuitPython Microcontroller Wiring

First wire up the breakout to your board exactly as follows. The CH9328 and the microcontroller will both be connected to your computer via USB but will share a ground connection. Make sure that the CH9328 onboard switches are set to Mode 3 (switch 2 off, switches 3 and 4 on) before powering it up. The following is the breakout wired to a Feather RP2040 using the JST SH connector.



Board GND to **breakout JST SH GND (black wire)**
Board TX to **breakout JST SH RX (white wire)**

The following is the breakout wired to a Feather RP2040 using a solderless breadboard:

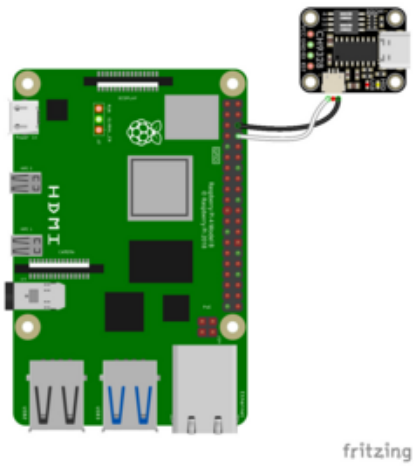


Board GND to **breakout GND (black wire)**
Board TX to **breakout RX (white wire)**

Python Computer Wiring

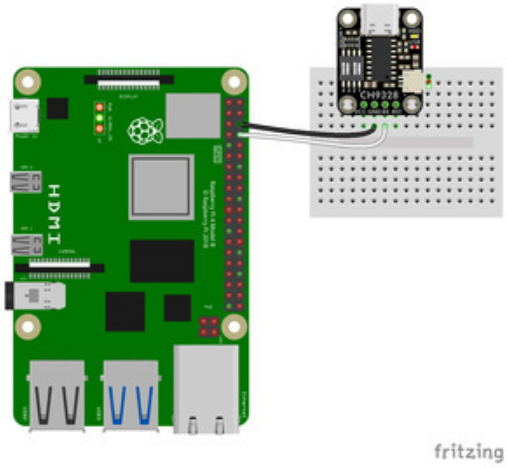
Since there are dozens of Linux computers/boards you can use, we will show wiring for Raspberry Pi. For other platforms, [please visit the guide for CircuitPython on Linux to see whether your platform is supported](https://adafru.it/BSN).

Here's the Raspberry Pi wired to the breakout using a JST SH cable:



Pi GND to **breakout JST SH GND (black wire)**
Pi TX to **breakout JST SH RX (white wire)**

Here is how you'll wire the breakout to a Raspberry Pi with a breadboard:



Pi GND to **breakout GND (black wire)**
Pi TX to **breakout RX (white wire)**

Python Installation of the Adafruit CH9328 Library

You'll need to install the **Adafruit_Blinka** library that provides the CircuitPython support in Python. This may also require enabling I2C and built-in UART (https://adafru.it/CEk) on your platform and verifying you are running Python 3. Since each platform is a little different, and Linux changes often, please visit the CircuitPython on Linux guide to get your computer ready (https://adafru.it/BSN)!

Once that's done, from your command line run the following command:

- `pip3 install adafruit-circuitpython-ch9328`

If your default Python is version 3 you may need to run 'pip' instead. Just make sure you aren't trying to use CircuitPython on Python 2.x, it isn't supported!

Make sure to follow the built-in UART setup steps if you are using the TX pin on the Raspberry Pi.

CircuitPython Usage

To use with CircuitPython, you need to first install the **Adafruit_CircuitPython_CH9328** library into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following folder:

- **adafruit_ch9328/**

Python Usage

Once you have the library `pip3` installed on your computer, copy or download the following example to your computer, and run the following, replacing **code.py** with whatever you named the file:

```
python3 code.py
```

Example Code

Once everything is saved to the **CIRCUITPY** drive, connect to the serial console (https://adafru.it/Bec) to see the data printed out!

```
# SPDX-FileCopyrightText: Copyright (c) 2024 Liz Clark for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""Simple demo to type "Hello World!" and then delete it"""

import time
import board
from adafruit_ch9328.ch9328 import Adafruit_CH9328
from adafruit_ch9328.ch9328_keymap import Keymap

# Initialize UART for the CH9328
# check for Raspberry Pi
# pylint: disable=simplifiable-condition
if "CE0" and "CE1" in dir(board):
    import serial

    uart = serial.Serial("/dev/ttyS0", baudrate=9600, timeout=3000)
# otherwise use busio
else:
    import busio

    uart = busio.UART(board.TX, board.RX, baudrate=9600)
ch9328 = Adafruit_CH9328(uart)

# Wait for 2 seconds
time.sleep(2)
# Sending "Hello World!" as an ASCII character string
ch9328.send_string("Hello World!")
# Wait for 2 seconds
time.sleep(2)

# Send the backspace key 12 times to erase the string
keys = [Keymap.BACKSPACE, 0, 0, 0, 0, 0]  # Keycode for backspace in US mapping
no_keys_pressed = [0, 0, 0, 0, 0, 0]
for _ in range(12):
    ch9328.send_key_press(keys, 0)  # Press
    ch9328.send_key_press(no_keys_pressed, 0)  # Release the key
```

In the code, the CH9328 is initialized over UART. After a two second delay, the text "Hello World!" is typed out from the CH9328. After an additional two second delay, the text is deleted. The delays are there so that you can position your cursor before the HID key presses are sent.

Python Docs

Python Docs (https://adafru.it/1a2A)

CPython

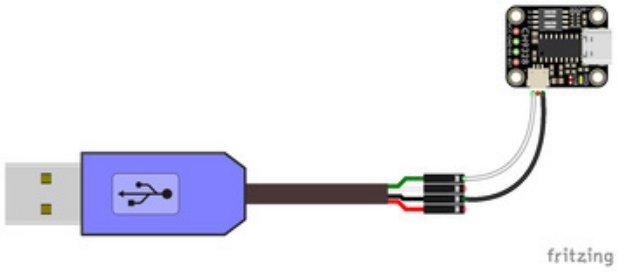
The CH9328 can allow you to do 'funky' things. In this example, you can have one desktop or single-board computer 'type' into a device such as a computer or mobile device by running our Python script and having it send UART data via a USB-to-UART converter (http://adafru.it/954).

USB to TTL Serial Cable - Debug / Console Cable for Raspberry Pi
The cable is easiest way ever to connect to your microcontroller/Raspberry Pi/WiFi router serial console port. Inside the big USB plug is a USB<->Serial conversion chip and at...
https://www.adafruit.com/product/954

Wiring

You'll plug the USB to TTL cable into the computer running the CPython script. The cable will connect to the CH9328 breakout RX and GND connections. The CH9328 breakout will plug into the computer that you want to send the HID commands to.

The USB to TTL cable has four wires: red power, black ground, white RX into USB port, and green TX out of the USB port. You'll be using the black ground wire and the green TX wire.



USB to TTL cable GND to **breakout JST SH GND (black wire)**
USB to TTL cable TX to **breakout JST SH RX (white wire)**

The CH9328 should be in Mode 0. Switches 2, 3 and 4 should all be switched on. The mode needs to be selected before the breakout is powered on.

Make sure that the CH9328 onboard switches are set to Mode 0 (switch 2, 3 and 4 on) before powering it up.

CPython CH9328 Code

To run the script you will need a desktop or single board computer with Python 3 installed.

```
# SPDX-FileCopyrightText: 2024 Liz Clark for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import sys
import serial
import keyboard

port = '/dev/ttyUSB0'  # Replace with your actual serial port
```

```
# Define a mapping for special characters when shift is pressed
SHIFTED_KEYS = {
    '1': '!', '2': '@', '3': '#', '4': '$', '5': '%',
    '6': '^', '7': '&', '8': '*', '9': '(', '0': ')',
    '`': '~', '-': '_', '=': '+', '[': '{', ']': '}',
    '\\': '|', ';': ':', "'": '"', ',': '<', '.': '>',
    '/': '?'
}

def send_key(serial_port, key):
    """
    Send a key press to the CH9328 via UART.

    Parameters:
    serial_port (serial.Serial): The serial port connection.
    key (str): The key to send.
    """
    serial_port.write(key.encode('ascii'))
    serial_port.flush()

def send_empty_report(serial_port):
    """
    Send an empty HID report to reset the state of the device.

    Parameters:
    serial_port (serial.Serial): The serial port connection.
    """
    try:
        empty_report = bytearray([0] * 8)
        serial_port.write(empty_report)
        serial_port.flush()
    except serial.SerialException as e:
        print(f"Failed to send empty report: {e}")

def main():
    # Configure the serial connection
    baudrate = 9600  # Default baud rate for CH9328 in Mode 1
    timeout = 1

    with serial.Serial(port, baudrate, timeout=timeout) as ser:

        print("Listening for keyboard inputs. Press 'ESC' to exit.")

        def on_key_event(event):
            if event.event_type == 'down':
                key = event.name
                if len(key) == 1:  # Only process single character keys
                    if keyboard.is_pressed('shift'):  # Check if shift is pressed
                        key = SHIFTED_KEYS.get(key, key.upper())
                    send_key(ser, key)
                elif key == 'space':
                    send_key(ser, ' ')
                elif key == 'enter':
                    send_key(ser, '\n')
            send_empty_report(ser)

        # Hook the keyboard event
        keyboard.hook(on_key_event)

        # Wait for ESC to exit
        keyboard.wait('esc')

if __name__ == "__main__":
    main()
    sys.exit()
```

CPython Dependencies

First, you'll use `pip` to install the Python libraries required to run the script:

```
pip install pyserial
pip install keyboard
```

Some versions of RapsberryPi OS use the 'venv' virtual environment and may balk at installing packages at the root level. 'keyboard' needs to be run at root level (which is scary and dangerous, beware) If you run into the 'error externally-managed-environment` use this instead:

```
sudo pip install pyserial --break-system-packages
sudo pip install keyboard --break-system-packages
```

Customize the Script

Open the script in your preferred text editor or IDE. At the top of the code, you'll need to update `port` to match the COM port (https://adafru.it/19Nc) that your USB to TTL cable is plugged into. Otherwise, the script will not work.

```
port = '/dev/ttyUSB0'  # Replace with your actual serial port
```

Update the COM port before running the script!

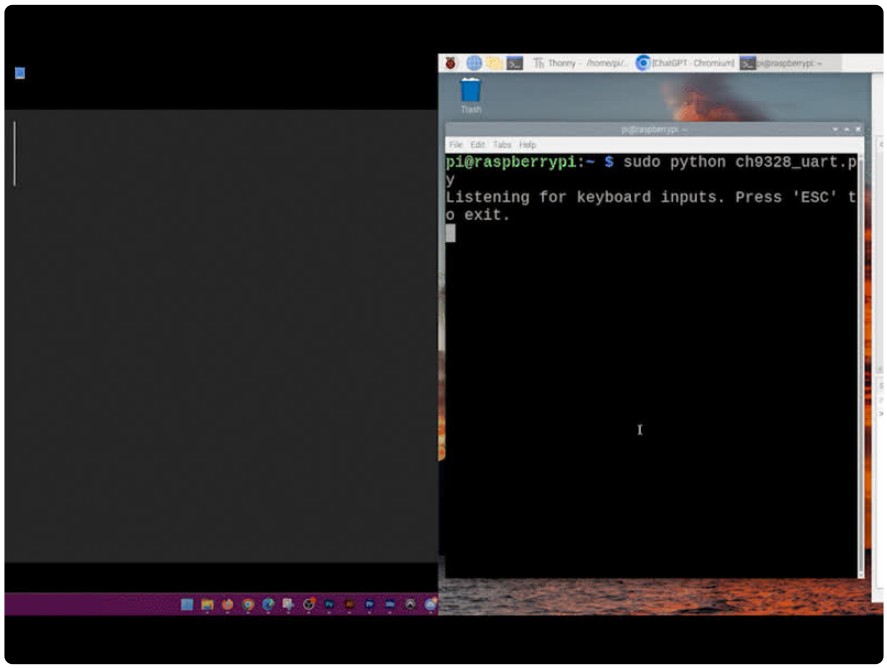
Run the Script

After adding your COM port, you can run the **code.py** script inside a terminal window on your computer with:

```
sudo python code.py
```

The `keyboard` library requires that the script be run as an administrator or `root`, depending on your operating system.

After the script starts to run, any keyboard typing that takes place on your computer will be mirrored via the USB TTL cable and CH9328 breakout. In the .GIF below, the script is running on a Raspberry Pi and the CH9328 is connected to a Windows PC.



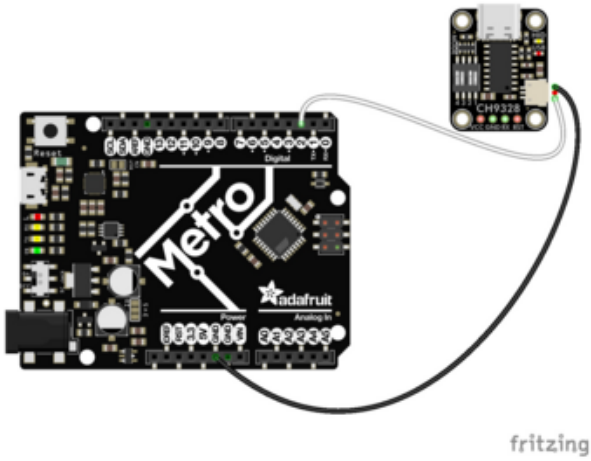
Arduino

Using the CH9328 breakout with Arduino involves wiring up the breakout to your Arduino-compatible microcontroller, installing the Adafruit_CH9328 (https://adafru.it/1a2H) library and running the provided example code.

Wiring

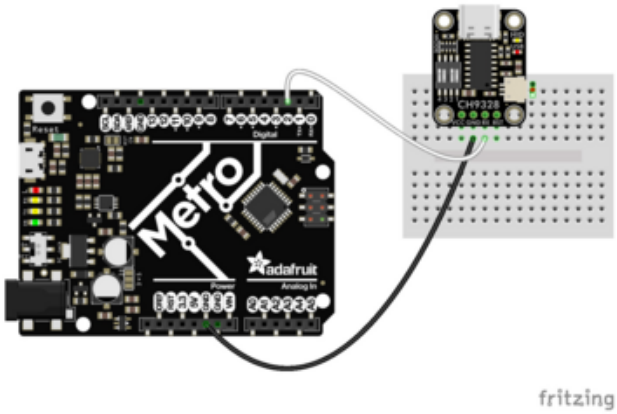
Wire as shown when you are uploading code to the microcontroller. The CH9328 and the microcontroller will both be connected to your computer via USB but will share a ground connection. Make sure that the CH9328 onboard switches are set to Mode 3 (switch 2 off, switches 3 and 4 on) before powering it up.

Here is an Adafruit Metro wired up to the breakout using a JST SH cable.



Board GND to **breakout JST SH GND (black wire)**
Board pin 2 to **breakout JST SH RX (white wire)**

Here is an Adafruit Metro wired up using a solderless breadboard:

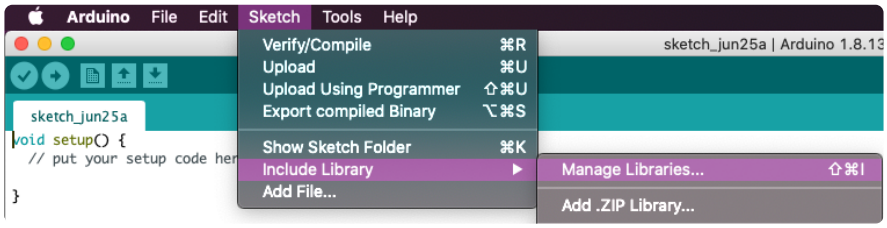


Board GND to **breakout GND (black wire)**
Board pin 2 to **breakout RX (white wire)**

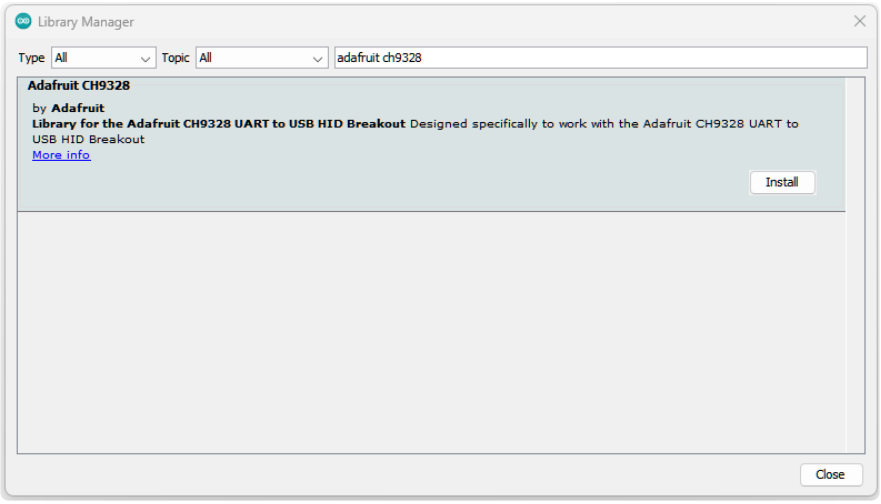
Make sure that the CH9328 onboard switches are set to Mode 3 (switch 2 off, switches 3 and 4 on) before powering it up.

Library Installation

You can install the Adafruit CH9328 library for Arduino using the Library Manager in the Arduino IDE.



Click the **Manage Libraries ...** menu item, search for **Adafruit CH9328**, and select the **Adafruit CH9328** library:



There are no additional dependencies for this library.

Example Code

```
/*
 *  This demo is for the "transparent transmission" mode #3 which you can enter by
 *  setting the Adafruit CH9328 switches so that #2 is OFF, and #3 and #4 are ON
 *  In this mode you can send 'raw' HID key commands for more complex keypress
combos!
 */
#include <Adafruit_CH9328.h>

Adafruit_CH9328 keyboard;

// Uncomment these lines if you need to use SoftwareSerial because there's no
Serial1 device
#if defined(__AVR_ATmega328P__)
  #include <SoftwareSerial.h>
  SoftwareSerial Serial1(3, 2); // RX, TX pins
#endif

void setup() {
  // Start the debug serial port, wait till it opens
  Serial.begin(115200);
  while (!Serial) delay(100);

  // The default baud rate for the CH9328 is 9600
  Serial1.begin(9600);
  keyboard.begin(&Serial1);

  // Sending "Hello World!" as an ASCII character string
  keyboard.typeString("Hello World!");

  // Wait for 1 second
  delay(1000);
```

```
  // Send the backspace key 12 times to erase the string
  byte keys[6] = {KEY_BACKSPACE, 0, 0, 0, 0, 0}; // Keycode for backspace in US
mapping
  byte noKeysPressed[6] = {0, 0, 0, 0, 0, 0};
  for (int i = 0; i < 12; i++) {
    keyboard.sendKeyPress(keys, 0);          // Press &
    keyboard.sendKeyPress(noKeysPressed, 0); // Release the key
  }
}

void loop() {
}
```

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. The text "`Hello World!`" will be typed out and then deleted. This will only happen once, since the loop in the example is empty.

Arduino Docs

Arduino Docs (https://adafru.it/1a2B)

Downloads

Files

- CH9328 Datasheet (https://adafru.it/1a2I)
- EagleCAD PCB Files on GitHub (https://adafru.it/1a2J)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/1a2K)

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for Interface Development Tools ***category:***

Click to view products by Adafruit ***manufacturer:***

Other Similar products are found below :

CY4607M CHA2066-99F XR17V358/SP339-E8-EB XR33052IDEVB P0551 5346 4901 LIME2-SHIELD PL2303 USB UART Board (type A) V2 MIKROE-5492 5964 TJA1145A-EVB CP2104-MINIEK 103030295 MIKROE-2335 KIT_MINIWIGGLER_3_USB KITXMC4XCOMETH001TOBO1 SI871XSOIC8-KIT 1764 1833 1862 ZSC31010KITV2.1 EVALISO1I811TTOBO1 EVB-USB82514 ATAB663231A-V1.2 2264 MCP23X17EV PS09-EVA-KIT FR12-0002 MAFR-000667-000001 MAFR-000589-000001 MAFR-000553-000001 BOB-13263 ORG4572-R01-UAR XR21B1422IL40-0A-EVB XR21B1420IL28-0A-EVB 28030 SKYFR-000743 SKYFR-000827 SKYFR-000982 MIKROE-2750 292 DFR0065 DFR0504 TEL0010 TEL0038 TEL0070 ASI4UEVBV2P1 NCN5193GEVB MIKROE-2517