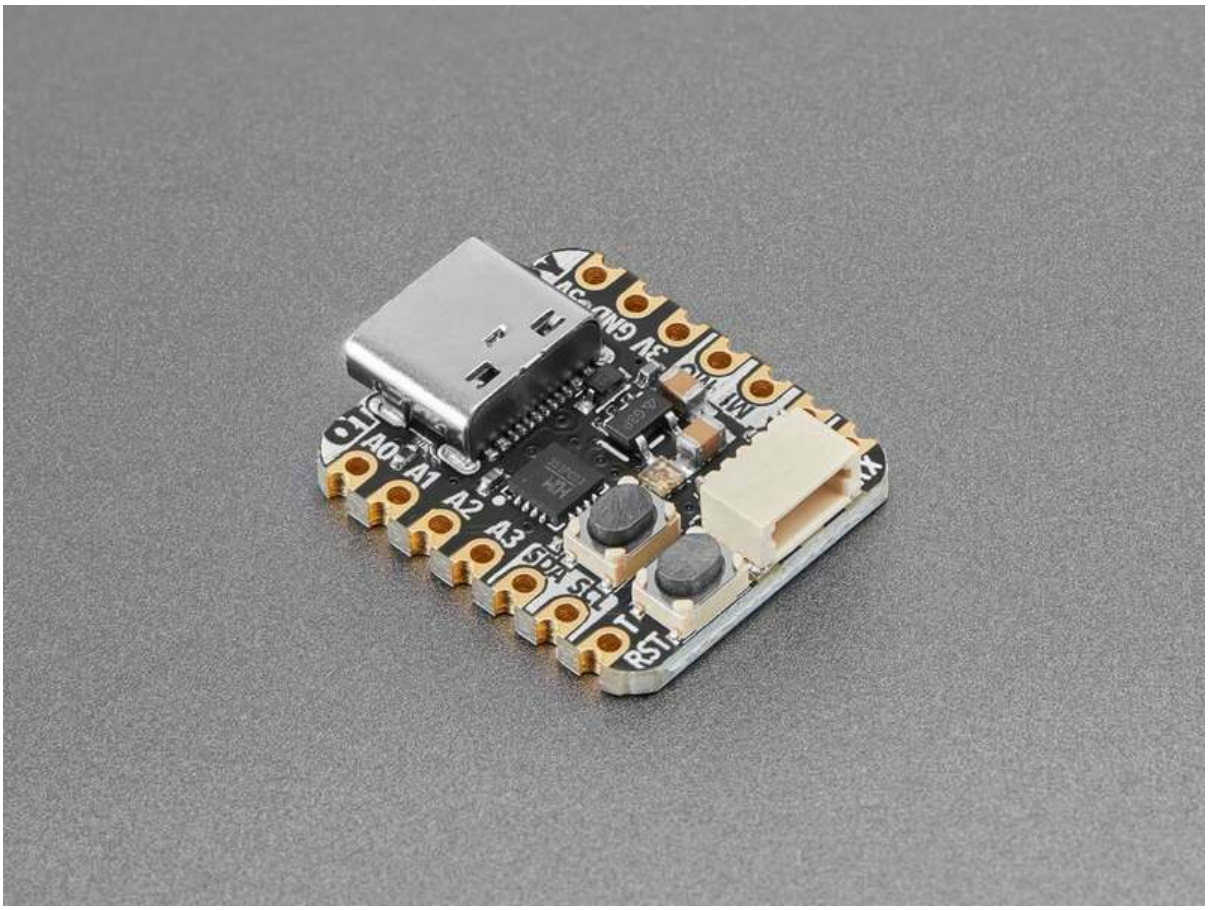
Adafruit CH552 QT Py

Created by Liz Clark

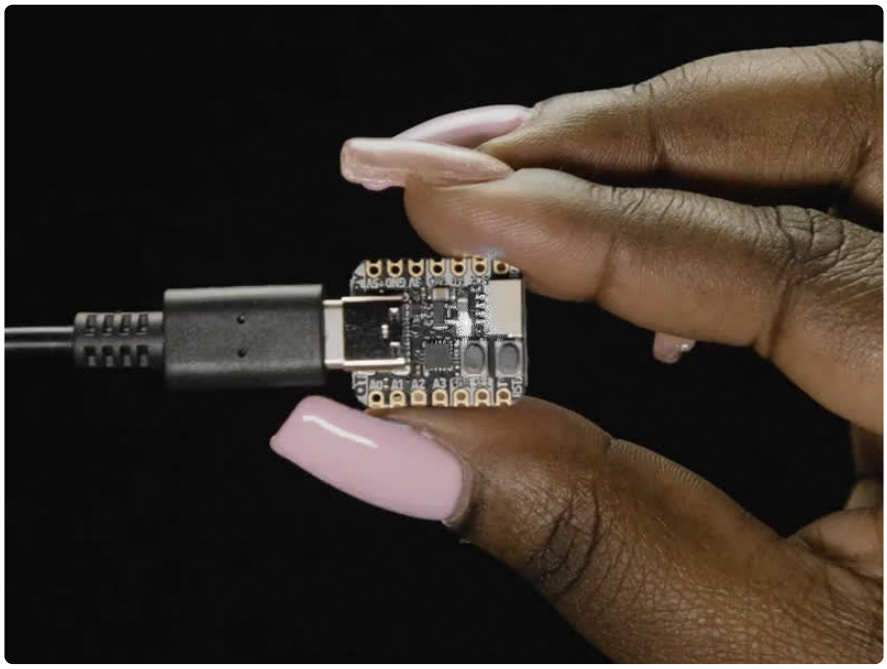


https://learn.adafruit.com/adafruit-ch552-qt-py

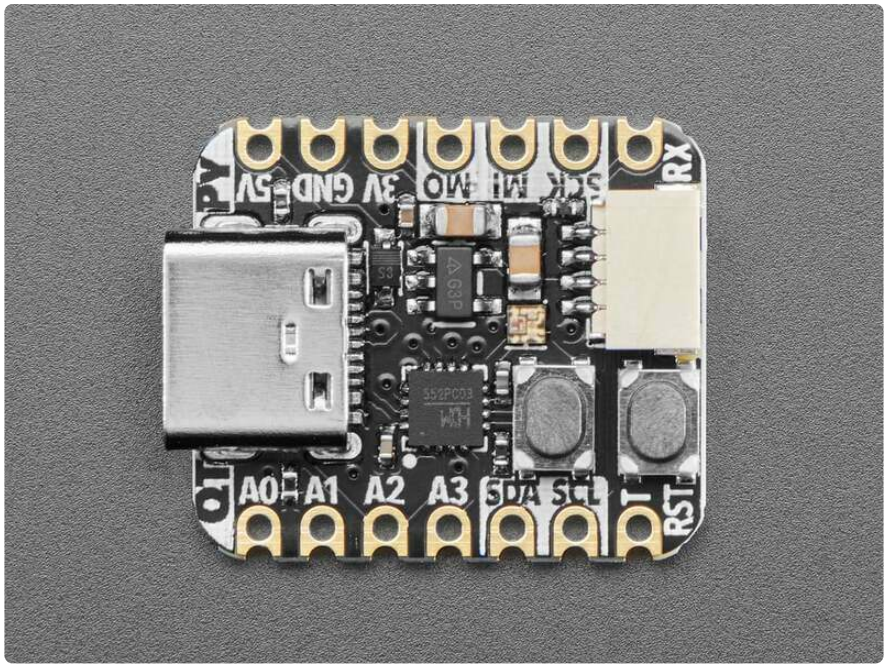
Last updated on 2025-01-23 04:07:54 PM EST

Table of Contents

Overview



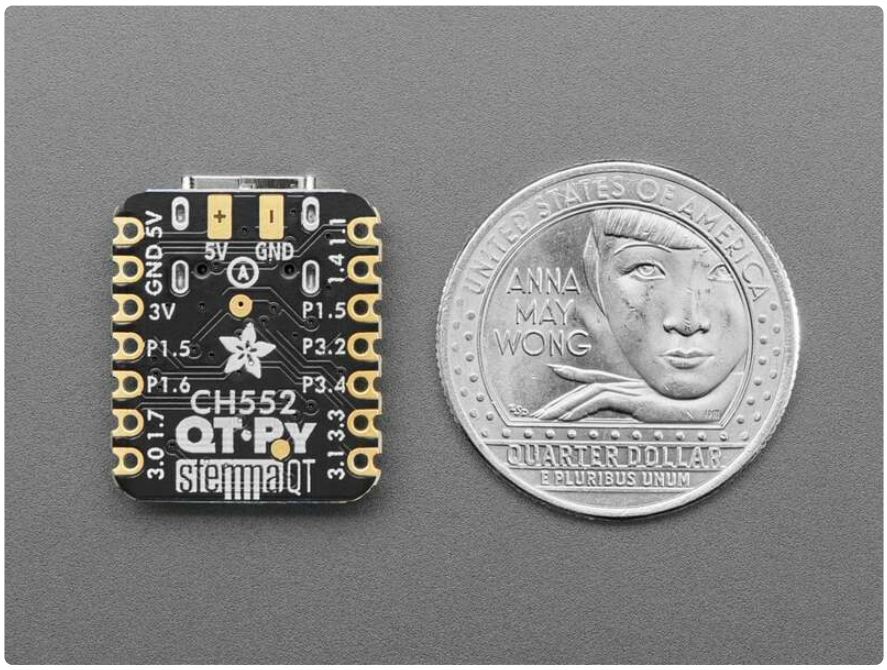
What a cutie pie! Or is it... a QT Py? This diminutive dev board comes with a throwback processor - an 8-bit 8051! This tiny core is a big change from something like the ESP32-S3 QT Py with two 240MHz 32-bit cores (http://adafru.it/5700), but there are lots of folks interested in the CH552 (https://adafru.it/19Zf) and, given the smol size, it is a nice matchup for a smol board.



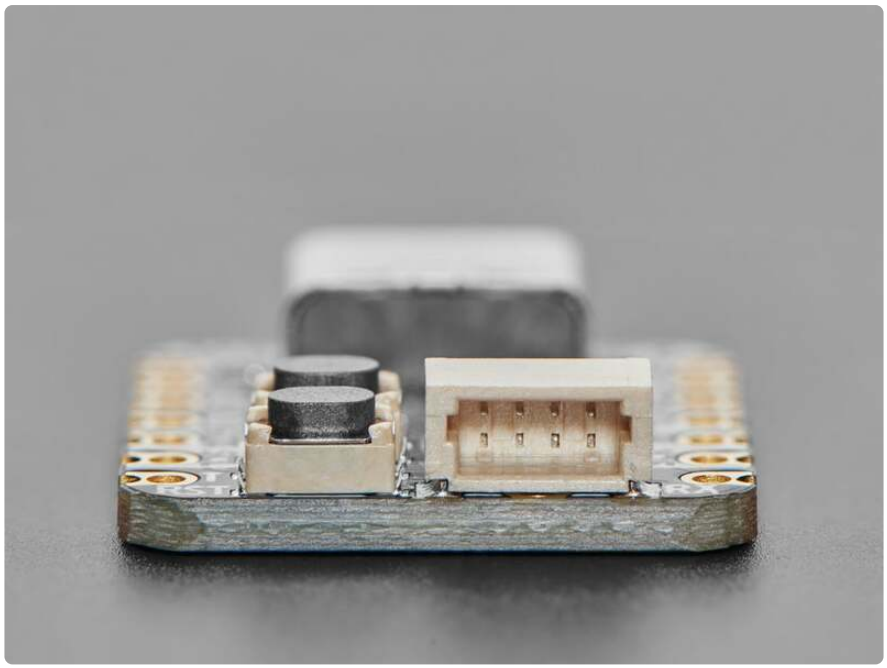
The CH552 is an 'enhanced' E8051 core microcontroller (https://adafru.it/19Zf), compatible with the MCS51 instruction set but with 8~15 times faster instruction execution speed. You can run this core at 16MHz and 3.3V logic, and it's got built-in

16K program FLASH memory and, 256-byte internal RAM plus 1K-byte internal xRAM (xRAM supports DMA.

It's also got some cute tricks up its sleeve, like 4 built-in ADC channels, capacitive touch support, 3 timers / PWM channels, hardware UART, SPI, and a full-speed USB device controller. The last one means it can act like a native USB device such as CDC serial or mouse/keyboard HID.



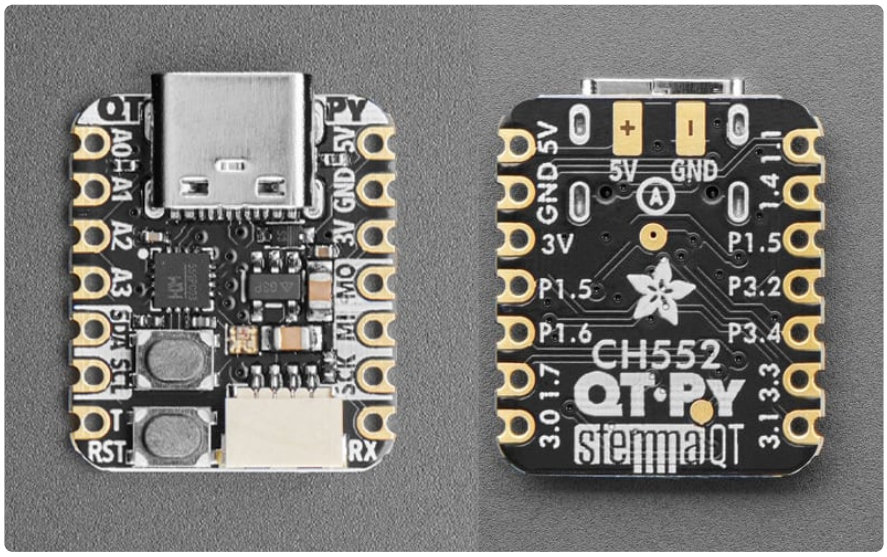
If you're interested in playing with this chip, we've wrapped it up in a QT Py format. The pinout and shape is Seeed Xiao (https://adafru.it/NC3) compatible, with castellated pads so you can solder it flat to a PCB. It comes with our favorite connector - the STEMMA QT (https://adafru.it/HMB), a chainable I2C port that can be used with any of our STEMMA QT sensors and accessories (https://adafru.it/NmD). We also added an RGB NeoPixel and both a reset button and 'bootloader enter' button.



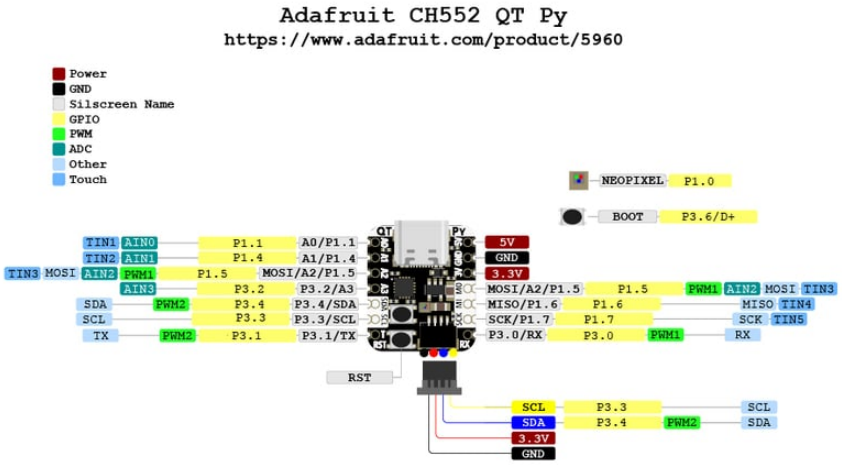
Please note! This is a minimal 8-bit microcontroller, and it definitely does not run CircuitPython or Micropython. It also doesn't really run Arduino. There's an Arduino 'board support package' (https://adafru.it/19ZA) we recommend, but the compiler is for C not C++, which means you cannot use any Arduino libraries. It's very very bare-bones and for hacking/experimenting with this '40 cent chip' (https://adafru.it/19ZB).

- It is the same size, form-factor, and pinout as the Seeed Xiao.
- **USB Type C connector** - If you have only Micro B cables, this adapter will come in handy (http://adafru.it/4299)!
- **CH552 8-bit 8051** microcontroller core with 3.3V power/logic. Internal 16 MHz oscillator.
- Native USB
- **Built in RGB NeoPixel LED**
- **10 GPIO pins:**
    - 4x 8-bit analog inputs on A0, A1, A2, and A3
    - 3 x PWM outputs
    - I2C port with STEMMA QT plug-n-play connector
    - Hardware UART
    - Hardware SPI
    - 4 x Capacitive Touch with no additional components required, on A0-A3 pins
- 3.3V regulator with **600mA peak output** (https://adafru.it/NC4)
- **Reset switch and bootloader** for starting your project code over or entering USB ROM bootloader mode
- **Really really small**

Pinouts

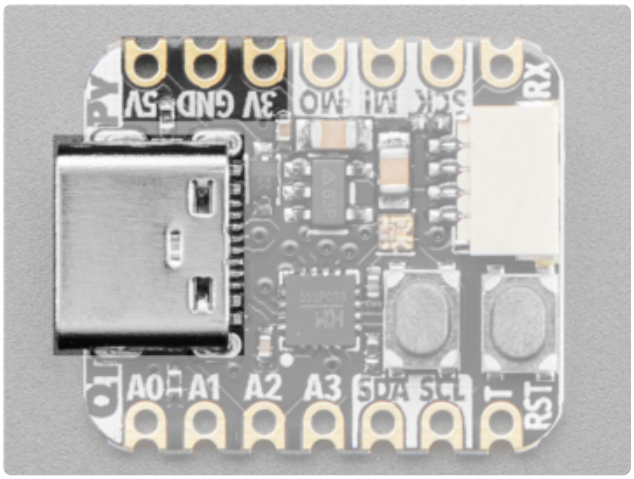


This QT Py does not run CircuitPython or MicroPython and cannot use any Arduino libraries because its compiler is for C, not C++.



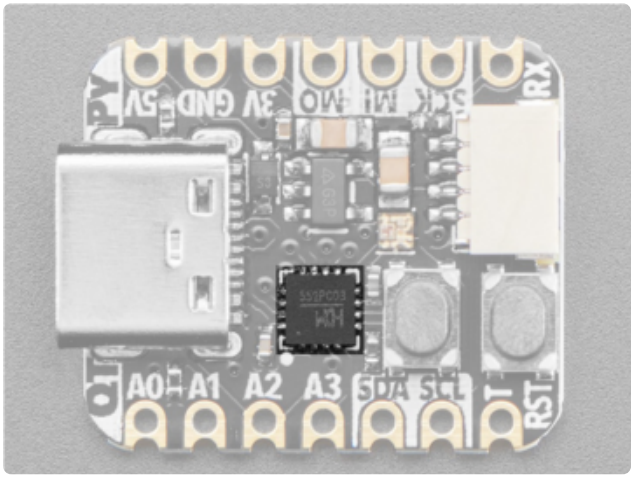
PrettyPins PDF on GitHub (https://adafru.it/19ZC).

Power



USB-C port - This is used for both powering and programming the board. You can power it with any USB C cable.
3V - This pin is the output from the 3.3V regulator (https://adafru.it/NC4), it can supply 600mA peak.
GND - This is the common ground for all power and logic.
5V - This is 5V out from the USB port.

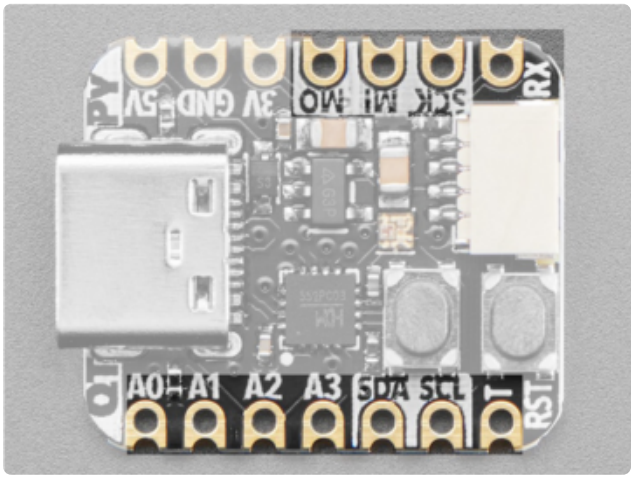
CH552 Chip



The CH552 is an 'enhanced' E8051 core microcontroller, compatible with the MCS51 instruction set but with 8 to 15 times faster instruction execution speed. You can run this core at 16MHz and 3.3V logic. It has the following features:

16K program FLASH memory
256-byte internal RAM
1K-byte internal xRAM (xRAM supports DMA)
4 built-in ADC channels
Capacitive touch support
2 timers / PWM channels
UART
SPI
Full-speed USB device controller

Logic Pins



There are ten GPIO pins broken out to pins. There is hardware I2C, UART, and SPI. **Note that A2 and MOSI share the same pin (P1.5).**

Four pins are 8-bit analog inputs (A0, A1, A2 and A3).

You can do PWM output on four of the pins (A2/MOSI, SDA, RX and TX).

There are five pins (A0, A1, A2/MOSI, A3, MISO and SCK) that can do capacitive touch without any external components needed.

I2C

Note that the CH552 does not have a 'native' I2C peripheral, so in CH55xduino this is bit-banged. However, we will call these the I2C pins

- **SCL** - This is the I2C clock pin. There is no pull-up on this pin, so for I2C please add an external pull-up if the breakout doesn't have one already. It's connected to P3.3.
- **SDA** - This is the I2C data pin. There is no pull-up on this pin, so for I2C please add an external pull-up if the breakout doesn't have one already. It's connected to P3.4.

These pins are also connected to the STEMMA QT port.

UART

- **RX** - This is the UART receive pin. Connect to TX (transmit) pin on your sensor or breakout. It's connected to P3.0.
- **TX** - This is the UART transmit pin. Connect to RX (receive) pin on your sensor or breakout. It's connected to P3.1.

SPI

- **SCK** - This is the SPI clock pin. It's connected to P1.7.
- **MI** - This is the SPI **M**icrocontroller **I**n / **S**ensor **O**ut pin. It's connected to P1.6.

- **MO** - This is the SPI **M**icrocontroller **O**ut / **S**ensor **I**n pin. **Note that this pin is shared with A2!** This pin can do capacitive touch and is one of the four ADC inputs. It's connected to P1.5.

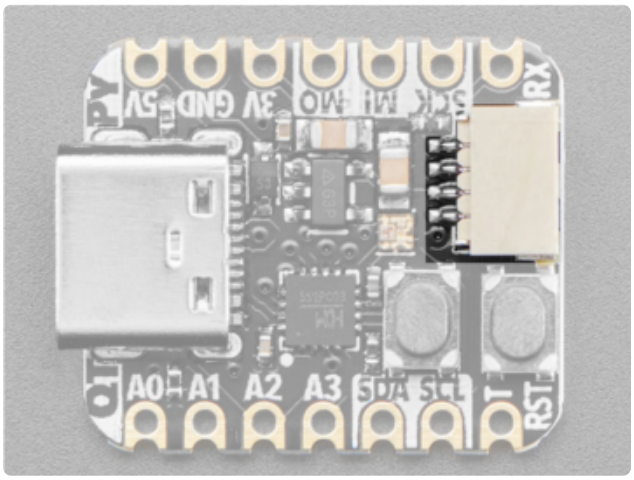
> The A2 pin is the same as MOSI pin.

Accessing Logic Pins with ch55xduino

The pins on the QT Py are accessed in the Arduino IDE by their GPIO number, minus the P and dot (.). For example, pin P1.0 is accessed as `10`. Pin P3.1 is accessed as `31`. Here is a list of all of the available pins on the QT Py as a `#define` list that you can include in your programs compiled with the ch55xduino BSP:

```
#define NEOPIXEL_PIN 10
#define A0 11
#define A1 14
#define A2 15
#define MOSI A2
#define MISO 16
#define SCK 17
#define RX 30
#define TX 31
#define A3 32
#define SCL 33
#define SDA 34
```

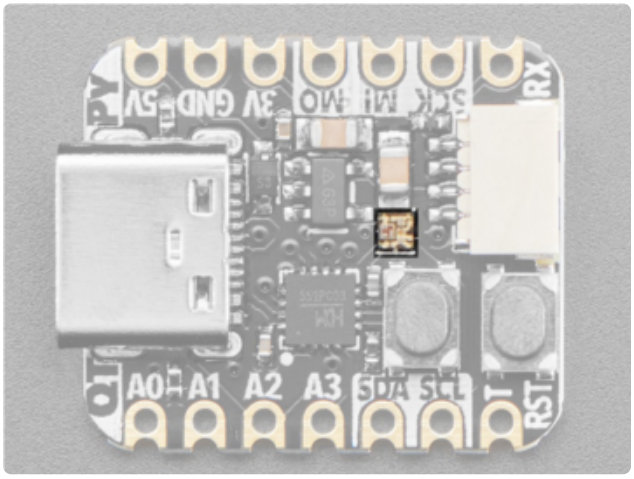
STEMMA QT Connector



This **JST SH 4-pin** STEMMA QT **(https://adafru.it/Ft4) connector** is located at the back of the board. It allows you to connect to various breakouts and sensors with **STEMMA QT** connectors (https://adafru.it/Qgf) or to other things using assorted associated accessories (https://adafru.it/Ft6). It works great with any STEMMA QT or Qwiic sensor/device. You can also use it with Grove I2C devices thanks to this handy cable (http://adafru.it/4528).

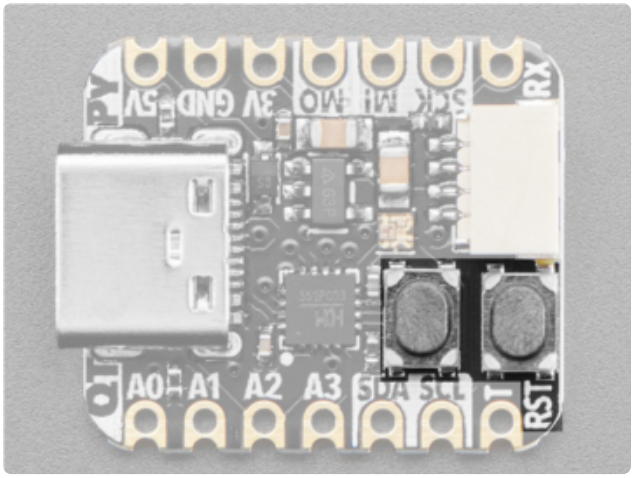
However, you **can't use this QT Py with any Arduino libraries** and it **does not run CircuitPython or MicroPython**. There's an Arduino 'board support package' (https://adafru.it/19ZA) we recommend, but the compiler is for C not C++.

NeoPixel LED



Next to the **BOOT** button, in the center of the board, is the **RGB NeoPixel LED**. This addressable LED can be controlled with code. It is connected to P1.0.

Buttons



Reset button - This button restarts the board and helps enter the bootloader. You can click it once to reset the board without unplugging the USB cable or battery.
BOOT button - This button is connected to P3.6/D+. To enter bootloader mode, disconnect the QT Py from USB power. Hold down the **BOOT** button and reconnect USB power.

Arduino IDE Setup

You've seen the warnings that you definitely can't use this QT Py with CircuitPython or MicroPython and that technically it doesn't work with Arduino either. What you can do though is use the ch55xduino board support package (https://adafru.it/19ZA) to write code in the Arduino IDE to compile with its C compiler.

Install Arduino IDE

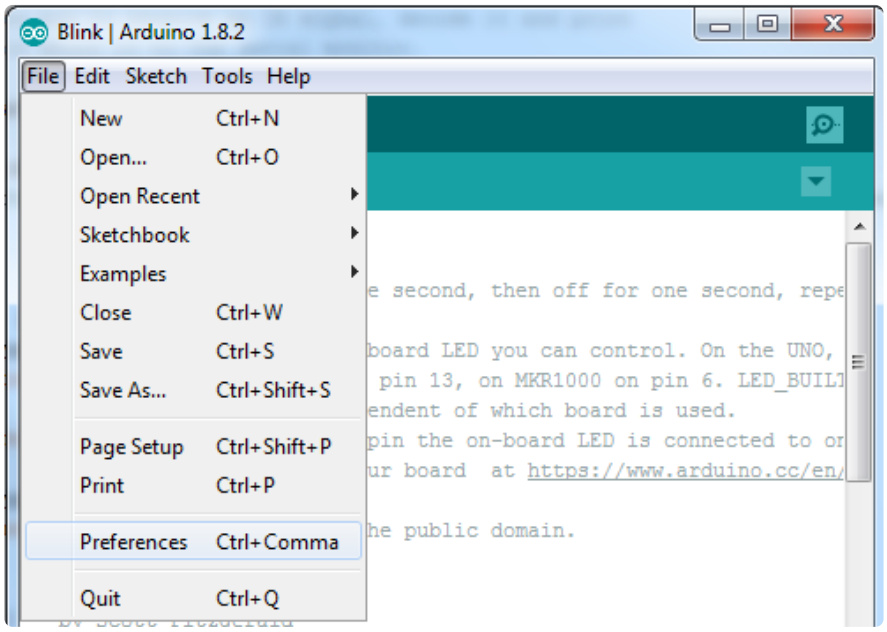
The first thing you will need to do is to download the latest release of the Arduino IDE. You will need to be using **version 1.8** or higher for this guide.

Arduino IDE Download

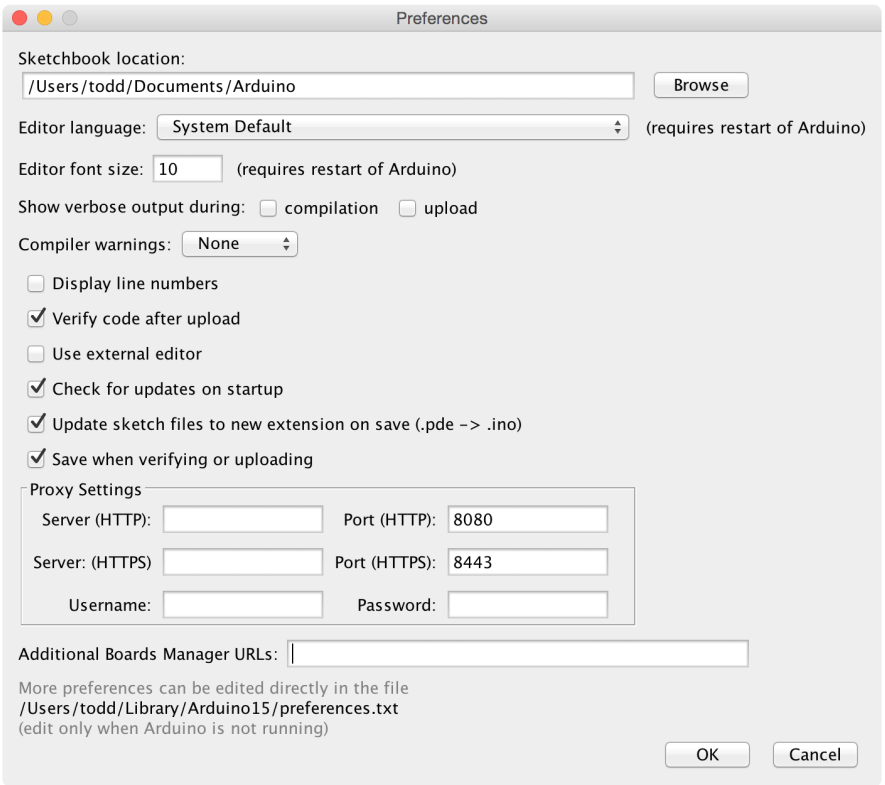
https://adafru.it/f1P

Install the ch55xduino Board Support Package

After you have downloaded and installed **the latest version of Arduino IDE**, you will need to start the IDE and navigate to the **Preferences** menu. You can access it from the **File** menu in Windows or Linux, or the **Arduino** menu on OS X.



A dialog will pop up just like the one shown below.

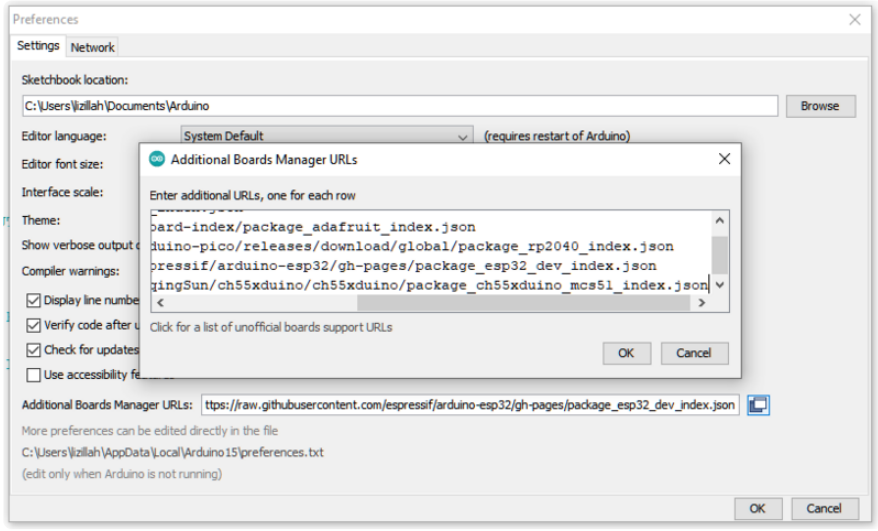


We will be adding a URL to the new **Additional Boards Manager URLs** option. The list of URLs is comma separated, and you will only have to add each URL once. New

Adafruit boards and updates to existing boards will automatically be picked up by the Board Manager each time it is opened. The URLs point to index files that the Board Manager uses to build the list of available & installed boards.

To find the most up to date list of URLs you can add, you can visit the list of third party board URLs on the Arduino IDE wiki (https://adafru.it/f7U). We will only need to add one URL to the IDE in this example, but **you can add multiple URLS by separating them with commas**. Copy and paste the link below into the **Additional Boards Manager URLs** option in the Arduino IDE preferences.

```
https://raw.githubusercontent.com/DeqingSun/ch55xduino/ch55xduino/
package_ch55xduino_mcs51_index.json
```

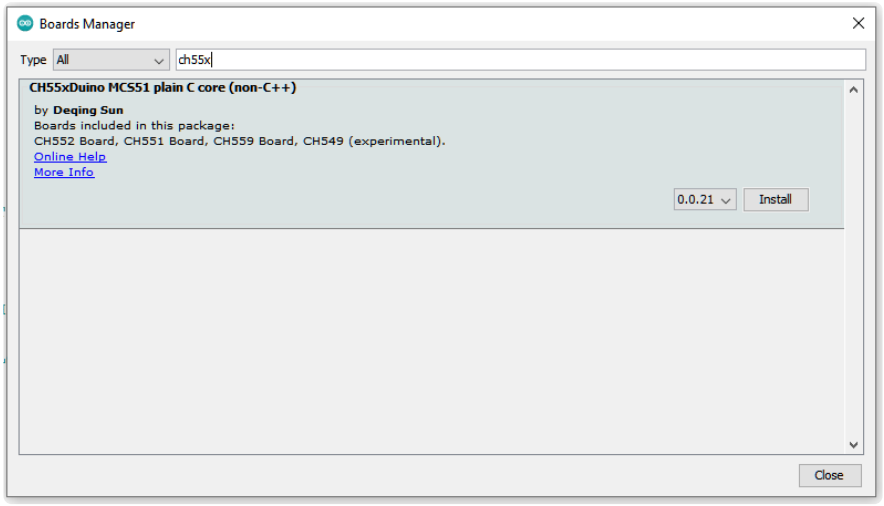


If you have multiple boards you want to support, say ESP8266 and Adafruit, have both URLs in the text box separated by a comma (,)

Once done click **OK** to save the new preference settings.

Install with the Board Manager

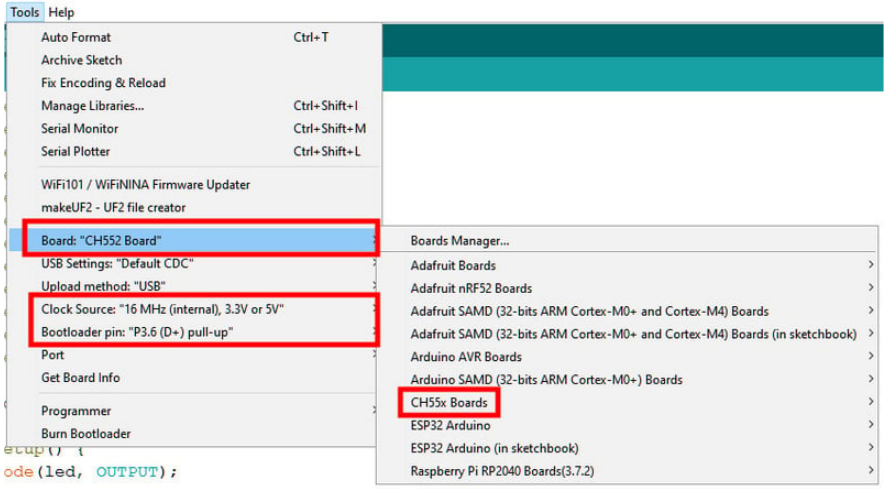
The next step is to actually install the Board Support Package (BSP). Go to the **Tools → Board → Board Manager** submenu. A dialog should come up with various BSPs. Search for **ch55xduino**.



Click the **Install** button and wait for it to finish. Once it is finished, you can close the dialog.

Code Upload Options

In the **Tools → Board** submenu you should see **CH55x Boards** and in that dropdown it should contain the CH55x boards.



Under **Board**, select **CH552 Board**. Under **Clock Source**, select **16 MHz (internal), 3.3V or 5V**. Under **Bootloader pin**, select **P3.6 (D+) pull-up**. These settings will work with the CH552 QT Py. Now you can try uploading some code examples to the board.

If you're not able to upload to the board, you may need to 'manually' put it into bootloader mode. Check this page https://learn.adafruit.com/adafruit-ch552-qt-py/bootloader-mode

Blink

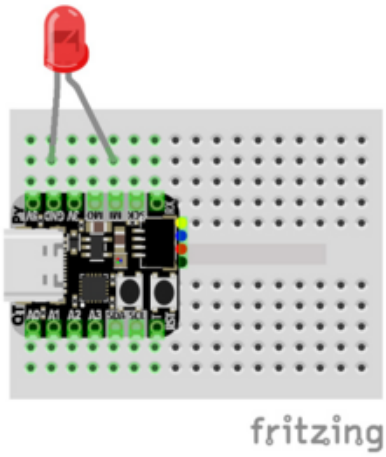
Blinking an LED is a great way to determine that you have your hardware and software ducks in a row. In this example, you'll breadboard an LED to the **MISO** pin (P1.6/ `16`), upload the example code and see your LED blink on and off every second.

Diffused 10mm LED Pack - 5 LEDs each in 5 Colors - 25 Pack

Need some chunky indicators? We are big fans of these diffused LEDs. They are fairly bright, so they can be seen in daytime, and from any angle. They go easily into a breadboard and...

https://www.adafruit.com/product/4204

Wiring



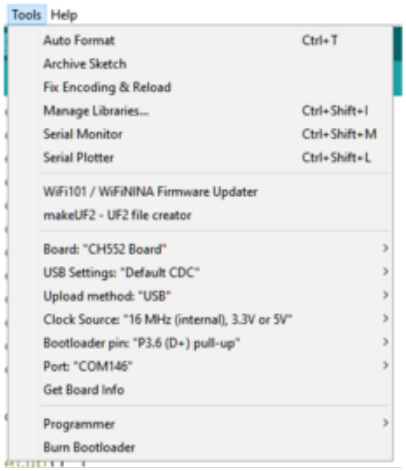
Board GND to **LED cathode (black wire)**

Board MISO to **LED anode (red wire)**

Blink Example

```
// SPDX-FileCopyrightText: 2024 ladyada for Adafruit Industries
//
// SPDX-License-Identifier: MIT

#define NEOPIXEL_PIN 10
#define A0 11
#define A1 14
#define A2 15
#define MOSI A2
#define MISO 16
#define SCK 17
#define RX 30
#define TX 31
#define A3 32
#define SCL 33
#define SDA 34
```

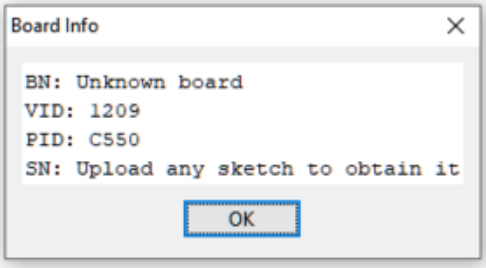
```
int led = MISO;

void setup() {
  pinMode(led, OUTPUT);
}

void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```



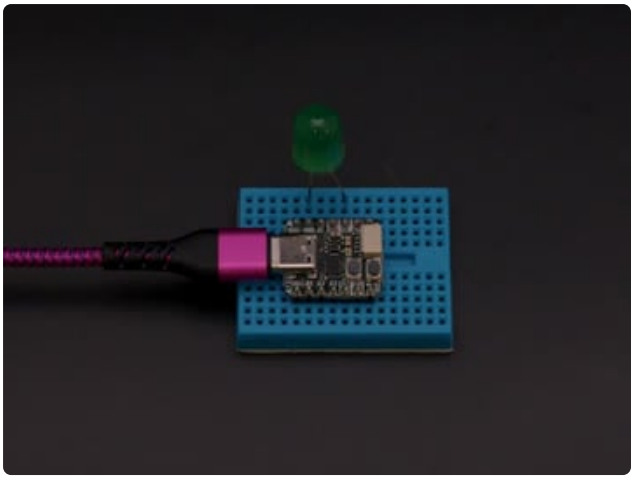
Confirm that your upload settings match the settings listed here under **Tools**:

Board: CH552 Board
USB Settings: Default CDC
Upload method: USB
Clock Source: 16 MHz (internal), 3.3V or 5V
Bootloader pin: P3.6 (D+) pull-up

For the port, select the COM port that matches your QT Py. It will not be labeled like you may be used to with other boards in the Arduino IDE.



You can confirm that you have the correct port selected by selecting **Get Board Info** from the Tools menu. This will open the **Board Info** window. The CH552 QT Py VID is **1209** and the PID is **C550**.



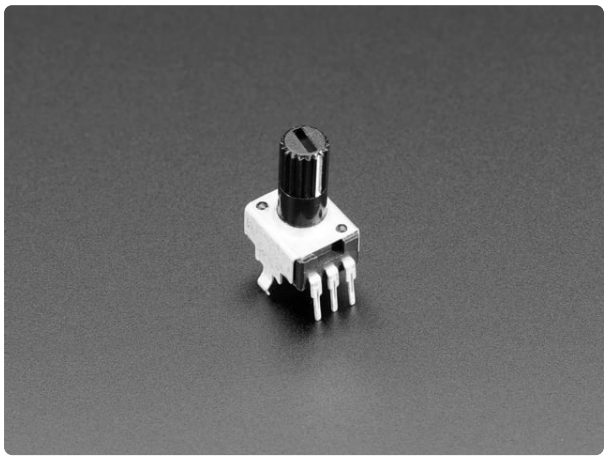
Upload the sketch to your board. You should see the LED blink on and off every second.

If you're not able to upload to the board, you may need to 'manually' put it into bootloader mode. Check this page https://learn.adafruit.com/adafruit-ch552-qt-py/bootloader-mode

---

Analog In

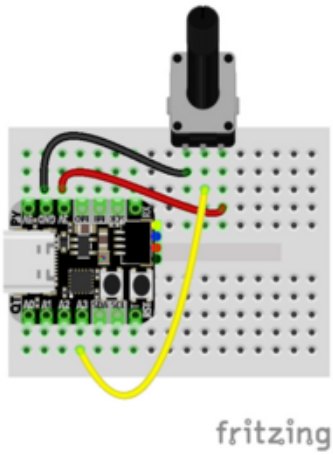
You can use one of the ADC pins on the QT Py as an analog input. In this example, you'll connect a potentiometer to pin **A3**, upload the example code to the board and use the Serial Monitor or Serial Plotter to see the signal on the pin fluctuate as you turn the potentiometer.



Potentiometer with Built In Knob - 10K ohm

Oh say can you seeBy the knob's early light...Sorry - we thought that was clever. And while it wasn't really, this potentiometer definitely...
https://www.adafruit.com/product/4133

Wiring

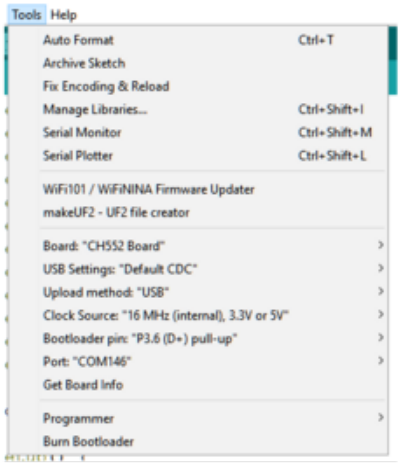


Board GND to **potentiometer GND (black wire)**
Board A3 to **potentiometer wiper (yellow wire)**
Board 3V to **potentiometer positive (red wire)**

Analog In Example

```
// SPDX-FileCopyrightText: 2024 ladyada for Adafruit Industries
//
// SPDX-License-Identifier: MIT

/*
  ReadAnalogVoltage

  Reads an analog input on pin P1.1, converts it to voltage, and prints the result
to the Serial Monitor.
  Graphical representation is available using Serial Plotter (Tools > Serial
Plotter menu).
  Attach the center pin of a potentiometer to pin P1.1, and the outside pins to +5V
and ground.

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/ReadAnalogVoltage
*/

#include <Serial.h>

#define A0 11
#define A1 14
#define A2 15 // also MISO!
#define A3 32

#define ANALOG_IN  A3
#define VREF       3.3

// the setup routine runs once when you press reset:
void setup() {
  // No need to init USBSerial

  // By default 8051 enable every pin's pull up resistor. Disable pull-up to get
full input range.
  pinMode(ANALOG_IN, INPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0, P1.1:
  int sensorValue = analogRead(ANALOG_IN);
  // Convert the analog reading (which goes from 0 - 255) to VREF:
  float voltage = sensorValue * (VREF / 255.0);
  // print out the value you read:
```

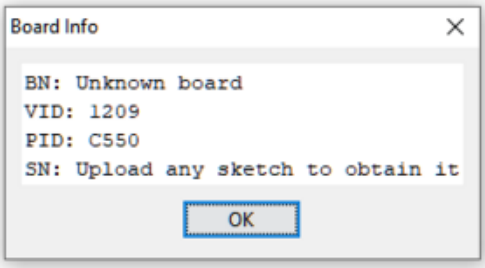
```
  USBSerial_println(voltage);
  // or with precision:
  //USBSerial_println(voltage,1);

  delay(10);
}
```



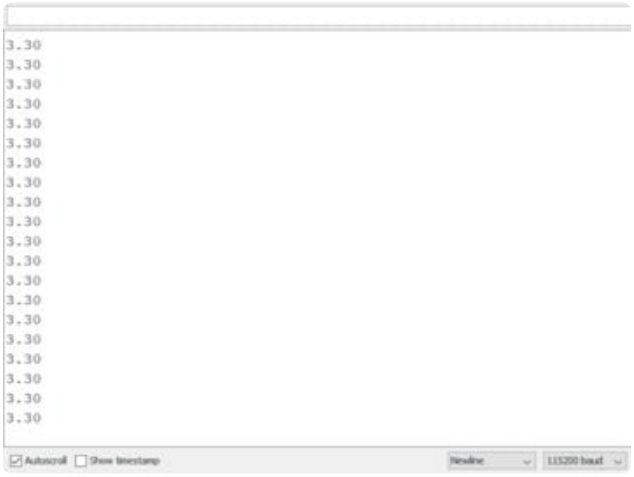
Confirm that your upload settings match the settings listed here under **Tools**:

Board: CH552 Board
USB Settings: Default CDC
Upload method: USB
Clock Source: 16 MHz (internal), 3.3V or 5V
Bootloader pin: P3.6 (D+) pull-up

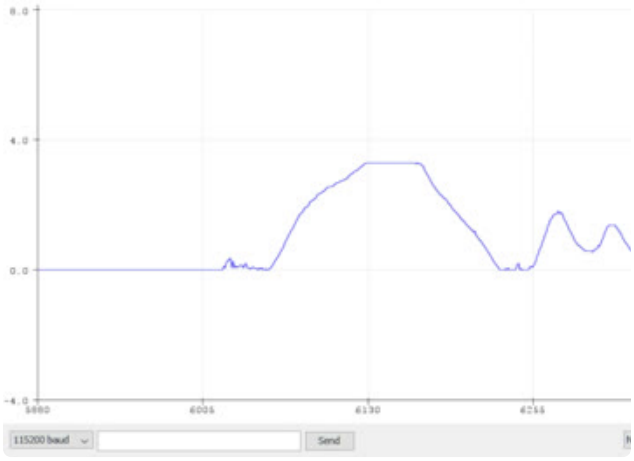
For the port, select the COM port that matches your QT Py. It will not be labeled like you may be used to with other boards in the Arduino IDE.



You can confirm that you have the correct port selected by selecting **Get Board Info** from the Tools menu. This will open the **Board Info** window. The CH552 QT Py VID is **1209** and the PID is **C550**.



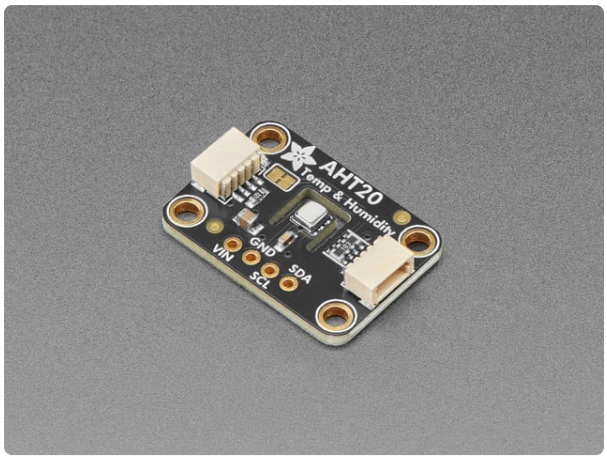
Upload the sketch to your board. Open the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. As you turn the potentiometer, you'll see the voltage reading on pin **A3** change.



For a more visual representation, you can open the Serial Plotter (**Tools -> Serial Plotter**) at 115200 baud. As you turn the potentiometer, the plotter will smoothly plot the voltage reading.

I2C

You can't use your favorite Arduino libraries with this board, but that doesn't mean you can't use your favorite I2C sensor. In this example, you'll connect an AHT20 temperature and humidity sensor the QT Py. Then, you'll upload the example code and open the Serial Monitor to see the temperature and humidity data print out.



Adafruit AHT20 - Temperature & Humidity Sensor Breakout Board

The AHT20 is a nice but inexpensive temperature and humidity sensor from the same folks that brought us the DHT22. You can take...

https://www.adafruit.com/product/4566

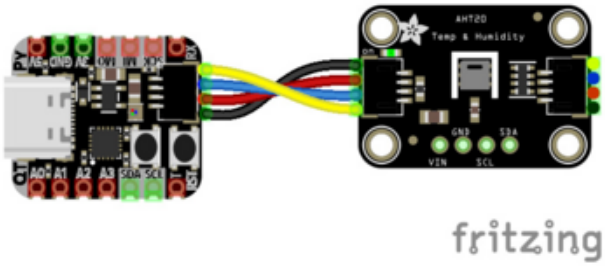
STEMMA QT / Qwiic JST SH 4-pin Cable - 100mm Long

This 4-wire cable is a little over 100mm / 4" long and fitted with JST-SH female 4-pin connectors on both ends. Compared with the chunkier JST-PH these are 1mm pitch instead of...

https://www.adafruit.com/product/4210

Wiring

You can connect the AHT20 sensor to the STEMMA QT port on the QT Py with a STEMMA QT cable.



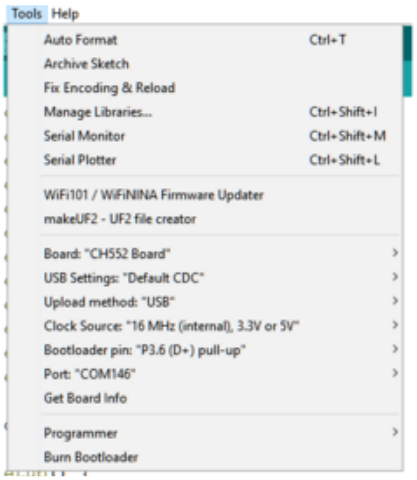
Board STEMMA GND to **sensor GND (black wire)**
Board STEMMA 3.3V to **sensor 3.3V (red wire)**
Board STEMMA SCL to **sensor SCL (yellow wire)**
Board STEMMA SDA to **sensor SDA (blue wire)**

AHT20 I2C Example

```
// SPDX-FileCopyrightText: 2024 ladyada for Adafruit Industries
//
// SPDX-License-Identifier: MIT
// https://chat.openai.com/share/5dddee44-3196-4a6b-b445-58ac6ef18501

#include <SoftI2C.h>

extern uint8_t scl_pin;
extern uint8_t sda_pin;

void Wire_begin(uint8_t scl, uint8_t sda);
bool Wire_scan(uint8_t i2caddr);
bool Wire_writeBytes(uint8_t i2caddr, uint8_t *data, uint8_t bytes);
bool Wire_readBytes(uint8_t i2caddr, uint8_t *data, uint8_t bytes);
bool Wire_readRegister(uint8_t i2caddr, uint8_t regaddr, uint8_t *data, uint8_t
bytes);

bool readAHT20(float *temperature, float *humidity);
#define AHTX0_I2CADDR_DEFAULT 0x38

void setup() {
  while (!USBSerial()); // wait for serial port to connect. Needed for native USB
port only
  delay(100);

  USBSerial_println("CH552 QT Py I2C sensor test");
  Wire_begin(33, 34); // set up I2C on CH552 QT Py

  USBSerial_print("I2C Scan: ");
  for (uint8_t a=0; a<=0x7F; a++) {
    if (!Wire_scan(a)) continue;
    USBSerial_print("0x");
    USBSerial_print(a, HEX);
    USBSerial_print(", ");
  }
  USBSerial_println();

  if (! Wire_scan(AHTX0_I2CADDR_DEFAULT)) {
    USBSerial_println("No AHT20 found!");
    while (1);
```

```
  }
}

void loop() {
  delay(100);

  float t, h;
  if (!readAHT20(&t, &h)) {
    USBSerial_println("Failed to read from AHT20");
  }
  USBSerial_print("Temp: ");
  USBSerial_print(t);
  USBSerial_print(" *C, Hum: ");
  USBSerial_print(h);
  USBSerial_println(" RH%");
}

/********************** AHT20 'driver */

#define AHTX0_CMD_TRIGGER 0xAC
#define AHTX0_STATUS_BUSY 0x80

bool AHT20_getStatus(uint8_t *status) {
  return Wire_readBytes(AHTX0_I2CADDR_DEFAULT, status, 1);
}

bool readAHT20(float *temperature, float *humidity) {
  uint8_t cmd[3] = {AHTX0_CMD_TRIGGER, 0x33, 0x00};
  uint8_t data[6], status;
  uint32_t rawHumidity, rawTemperature;

  // Trigger AHT20 measurement
  if (!Wire_writeBytes(AHTX0_I2CADDR_DEFAULT, cmd, 3)) {
    return false;
  }

  // Wait until the sensor is no longer busy
  do {
    if (!AHT20_getStatus(&status)) {
      return false;
    }
    delay(10); // Delay 10ms to wait for measurement
  } while (status & AHTX0_STATUS_BUSY);

  // Read the measurement data
  if (!Wire_readBytes(AHTX0_I2CADDR_DEFAULT, data, 6)) {
    return false;
  }

  // Parse humidity data
  rawHumidity = data[1];
  rawHumidity = (rawHumidity << 8) | data[2];
  rawHumidity = (rawHumidity << 4) | (data[3] >> 4);
  *humidity = ((float)rawHumidity * 100.0) / 0x100000;

  // Parse temperature data
  rawTemperature = (data[3] & 0x0F);
  rawTemperature = (rawTemperature << 8) | data[4];
  rawTemperature = (rawTemperature << 8) | data[5];
  *temperature = ((float)rawTemperature * 200.0 / 0x100000) - 50.0;

  return true;
}

/*************************** Wire I2C interface */

void Wire_begin(uint8_t scl, uint8_t sda) {
  scl_pin = scl; //extern variable in SoftI2C.h
  sda_pin = sda;
```

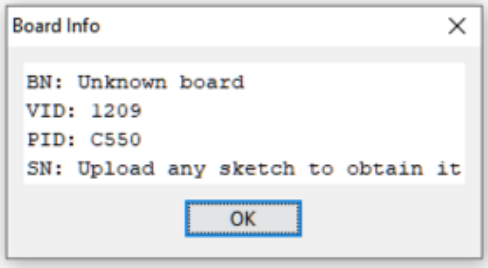
```
  I2CInit();
}

bool Wire_scan(uint8_t i2caddr) {
  return Wire_writeBytes(i2caddr, NULL, 0);
}

bool Wire_readRegister(uint8_t i2caddr, uint8_t regaddr, uint8_t *data, uint8_t
bytes) {
  if (!Wire_writeBytes(i2caddr, &regaddr, 1)) {
    return false;
  }

  return Wire_readBytes(i2caddr, data, bytes);
}


bool Wire_writeBytes(uint8_t i2caddr, uint8_t *data, uint8_t bytes) {
  uint8_t ack_bit;

  I2CStart();
  ack_bit = I2CSend(i2caddr << 1 | 0); // Shift address and append write bit
  if (ack_bit != 0) {
    I2CStop();
    return false;
  }

  for (uint8_t i = 0; i < bytes; i++) {
    if (I2CSend(data[i]) != 0) {
      I2CStop();
      return false;
    }
  }
  I2CStop();
  return true;
}

bool Wire_readBytes(uint8_t i2caddr, uint8_t *data, uint8_t bytes) {
  uint8_t ack_bit;

  I2CStart();
  ack_bit = I2CSend(i2caddr << 1 | 1); // Shift address and append read bit
  if (ack_bit != 0) {
    I2CStop();
    return false;
  }

  for (uint8_t i = 0; i < bytes; i++) {
    data[i] = I2CRead();
    if (i == bytes - 1) {
        I2CNak();  // NAK on last byte
    } else {
        I2CAck();  // ACK on other bytes
    }
  }

  I2CStop();
  return true;
}
```



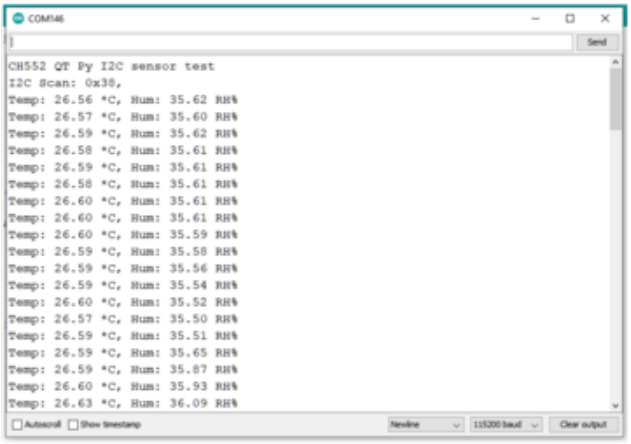
Confirm that your upload settings match the settings listed here under **Tools**:

Board: CH552 Board
USB Settings: Default CDC
Upload method: USB
Clock Source: 16 MHz (internal), 3.3V or 5V
Bootloader pin: P3.6 (D+) pull-up

For the port, select the COM port that matches your QT Py. It will not be labeled like you may be used to with other boards in the Arduino IDE.



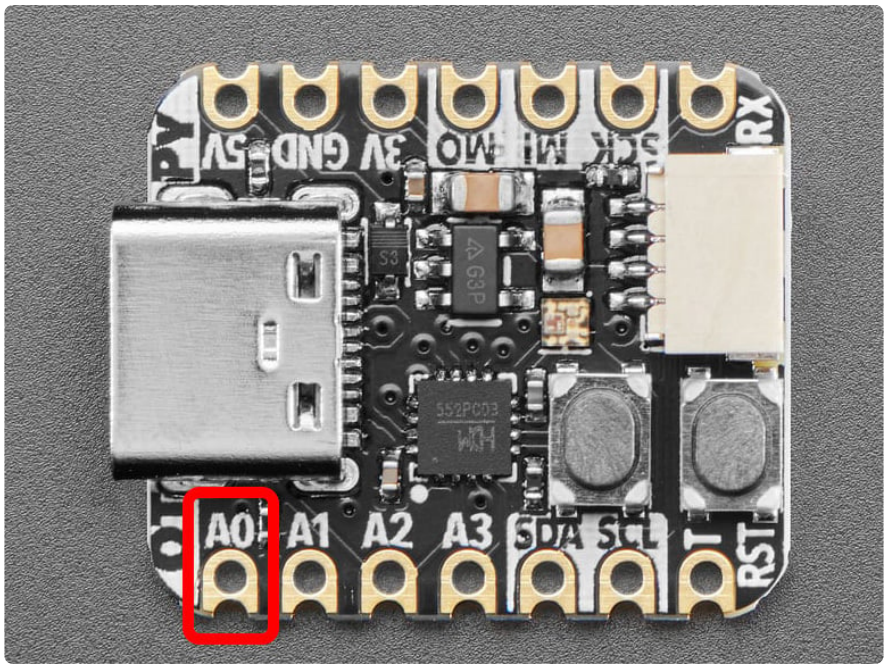
You can confirm that you have the correct port selected by selecting **Get Board Info** from the Tools menu. This will open the **Board Info** window. The CH552 QT Py VID is **1209** and the PID is **C550**.



Upload the sketch to your board. Open the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the temperature and humidity data print out.

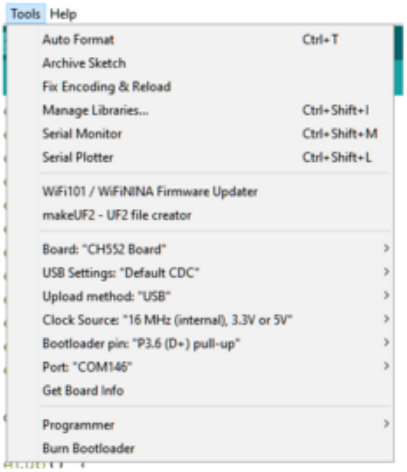
Capacitive Touch

The CH552 has capacitive touch support on a few pins (**A0, A1, A2/MOSI, A3, MISO** and **SCK**) without needing any external components. In this example, you'll upload the sketch to your board and use the Serial Monitor to monitor the touch input on pin **A0**.



Capacitive Touch Example

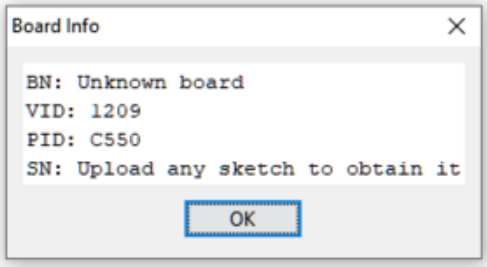
```
// SPDX-FileCopyrightText: 2024 ladyada for Adafruit Industries
//
// SPDX-License-Identifier: MIT

#include <TouchKey.h>

uint8_t count = 0;
uint8_t state = 0;

void setup() {
  while (!USBSerial()); // wait for serial port to connect. Needed for native USB
port only
  delay(100);
  USBSerial_println("QT Py CH552 Cap Touch Test");
  USBSerial_println("Uses pin A0 (P1.1)");
  TouchKey_begin((1 << 1)); //Enable channel P1.1/A0
}

void loop() {
  // put your main code here, to run repeatedly:
  TouchKey_Process();
  uint8_t touchResult = TouchKey_Get();
  if (touchResult) {
    if (state == 0) {
    count += 1;
    state = 1;
    USBSerial_print("TIN1.1 touched ");
    USBSerial_print(count);
    USBSerial_println(" times");
    }
  } else {
    state = 0;
  }
  delay(1000);

}
```



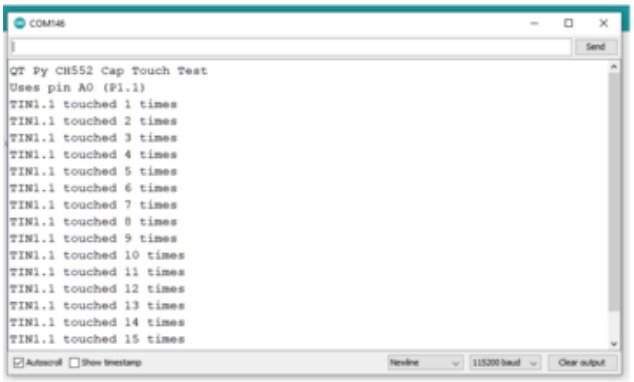
Confirm that your upload settings match the settings listed here under **Tools**:

Board: CH552 Board
USB Settings: Default CDC
Upload method: USB
Clock Source: 16 MHz (internal), 3.3V or 5V
Bootloader pin: P3.6 (D+) pull-up

For the port, select the COM port that matches your QT Py. It will not be labeled like you may be used to with other boards in the Arduino IDE.



You can confirm that you have the correct port selected by selecting **Get Board Info** from the Tools menu. This will open the **Board Info** window. The CH552 QT Py VID is **1209** and the PID is **C550**.

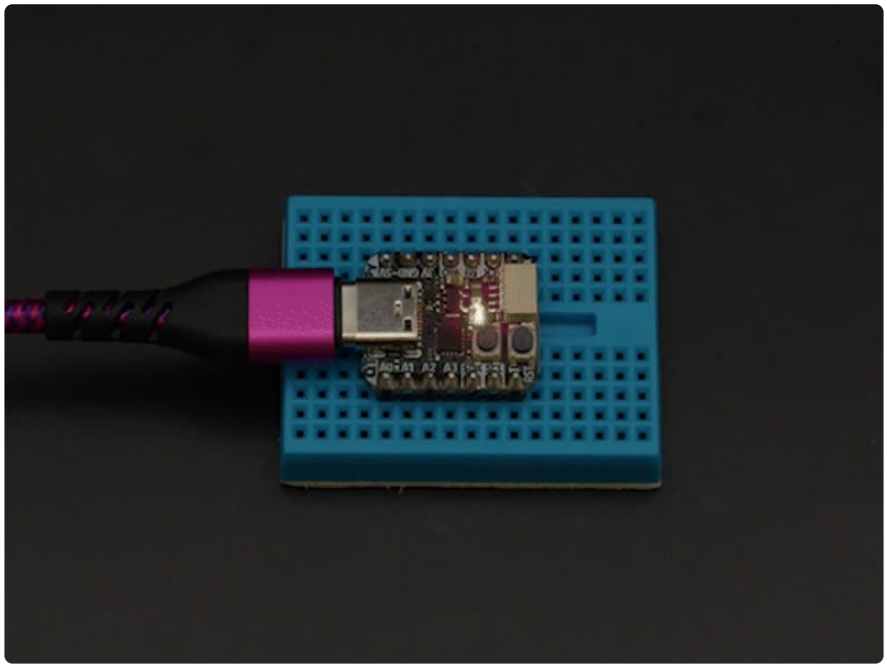


Upload the sketch to your board. Open the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. As you touch pin **A0**, you'll see a print out to the monitor. The `count` will increase by `1` with every touch.

NeoPixel

You can't use Arduino libraries with the CH552, so does that mean you can't use ever-so-colorful, always magical NeoPixels? Nope, you absolutely can! In this example,

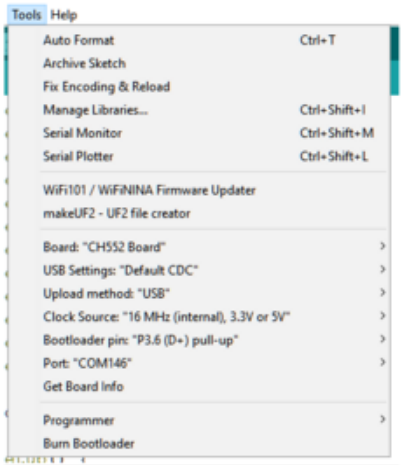
you'll upload the sketch to have the onboard NeoPixel (pin P1.0/`10`) perform a rainbow swirl.



NeoPixel Example

```
// SPDX-FileCopyrightText: 2024 ladyada for Adafruit Industries
//
// SPDX-License-Identifier: MIT

#include <WS2812.h>

#define NEOPIXEL_PIN P1_0
#define NUM_LEDS 1

#define COLOR_PER_LEDS 3
#define NUM_BYTES (NUM_LEDS*COLOR_PER_LEDS)
#if NUM_BYTES > 255
#error "NUM_BYTES can not be larger than 255."
#endif
__xdata uint8_t ledData[NUM_BYTES];

/*****************************************************************/
uint8_t neopixel_brightness = 255;
uint32_t Wheel(byte WheelPos);
void rainbowCycle(uint8_t wait);

#define NEOPIXEL_SHOW_FUNC CONCAT(neopixel_show_, NEOPIXEL_PIN)

void neopixel_begin() {
  pinMode(NEOPIXEL_PIN, OUTPUT); //Possible to use other pins.
}

void neopixel_show() {
  NEOPIXEL_SHOW_FUNC(ledData, NUM_BYTES); //Possible to use other pins.
}

void neopixel_setPixelColor(uint8_t i, uint32_t c) {
  uint16_t r, g, b;
  r = (((c >> 16) & 0xFF) * neopixel_brightness) >> 8;
  g = (((c >> 8) & 0xFF) * neopixel_brightness) >> 8;
  b = ((c & 0xFF) * neopixel_brightness) >> 8;
```

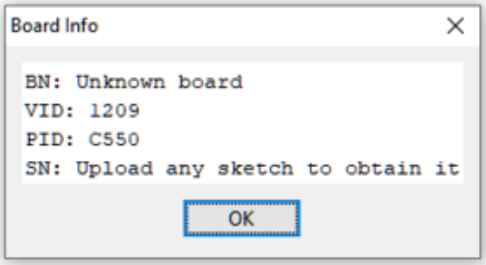
```
  set_pixel_for_GRB_LED(ledData, i, r, g, b);
}

void neopixel_setBrightness(uint8_t b) {
  neopixel_brightness = b;
}
/************************************************************************/


void setup() {
  neopixel_begin();
  neopixel_setBrightness(50);
}

void loop() {
  rainbowCycle(5);
}


void rainbowCycle(uint8_t wait) {
  uint8_t i, j;

  for (j=0; j<255; j++) {
    for (i=0; i < NUM_LEDS; i++) {
      neopixel_setPixelColor(i, Wheel(((i * 256 / NUM_LEDS) + j) & 255));
    }
    neopixel_show();
    delay(wait);
  }
}


// Input a value 0 to 255 to get a color value.
// The colours are a transition r - g - b - back to r.
uint32_t Wheel(byte WheelPos) {
  uint8_t r, g, b;
  uint32_t c;

  if(WheelPos < 85) {
   r = WheelPos * 3;
   g = 255 - WheelPos * 3 ;
   b = 0;
  } else if(WheelPos < 170) {
   WheelPos -= 85;
   r = 255 - WheelPos * 3;
   g = 0;
   b = WheelPos * 3;
  } else {
   WheelPos -= 170;
   r = 0;
   g = WheelPos * 3;
   b = 255 - WheelPos * 3;
  }
  c = r;
  c <<= 8;
  c |= g;
  c <<= 8;
  c |= b;
  return c;
}
```



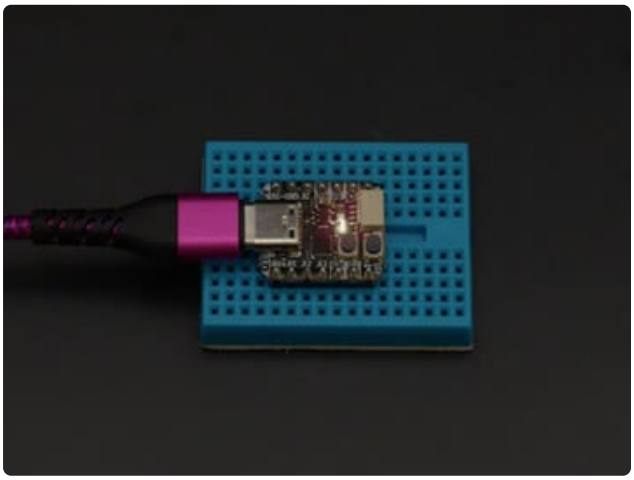
Confirm that your upload settings match the settings listed here under **Tools**:

Board: CH552 Board
USB Settings: Default CDC
Upload method: USB
Clock Source: 16 MHz (internal), 3.3V or 5V
Bootloader pin: P3.6 (D+) pull-up

For the port, select the COM port that matches your QT Py. It will not be labeled like you may be used to with other boards in the Arduino IDE.



You can confirm that you have the correct port selected by selecting **Get Board Info** from the Tools menu. This will open the **Board Info** window. The CH552 QT Py VID is **1209** and the PID is **C550**.



Upload the sketch to your board. You'll see the NeoPixel begin swirling thru the colors of the rainbow. This is the same demo that ships on the boards.

Manual Bootloader

Uploading directly to the QT Py USB port is very convenient. However, if you find that your COM port disappears or you're working on HID device code (https://adafru.it/19ZD), then you may need to utilize uploading code in bootloader mode.

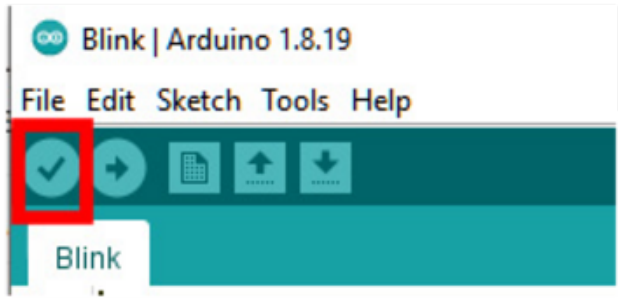
You can get your QT Py CH552 into bootloader mode by unplugging the USB port, holding down the **Boot** button and plugging the USB cable back in. There is a catch though: **the chip does not stay in bootloader mode for very long.** You only have a few seconds to talk to the bootloader. As a result, timing is everything.

Bootloader mode is the only way you can reprogram the chip after uploading HID code that is compiled with the `USER CODE` USB settings. It's also needed if bad code is uploaded that makes the COM port unreachable.

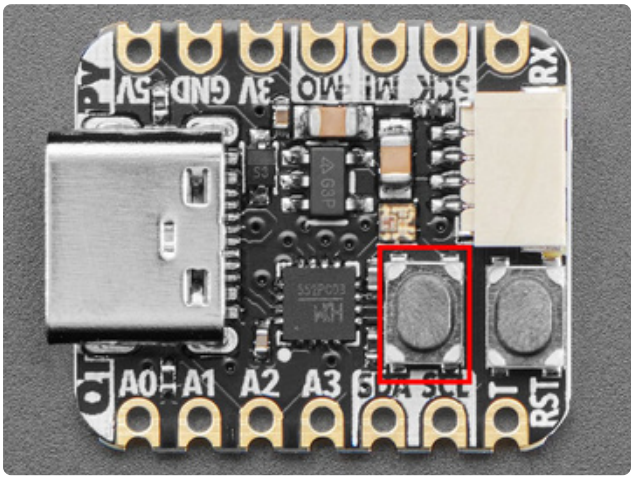
> The CH552 does not stay in bootloader mode for very long. You only have a few seconds to talk to the bootloader.

Blink to the Rescue

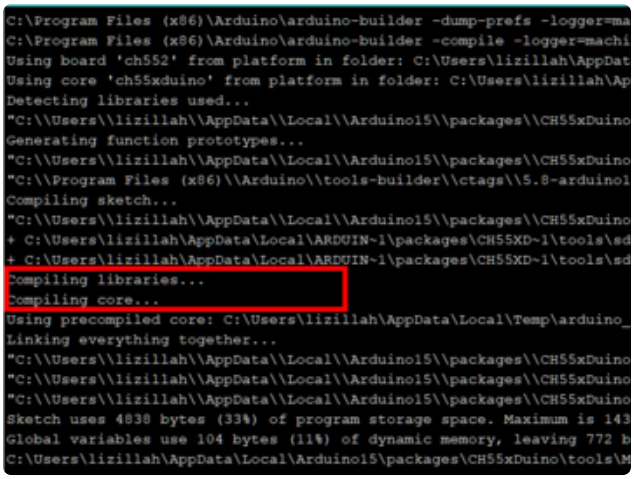
Blinking an LED is not only a great place to start with new hardware, it's also a great reset point; like a save state. You'll upload the blink example to the QT Py in bootloader mode.



After opening the sketch in the Arduino IDE, click on the **Verify checkmark** to compile the code. This will save some time when uploading the code to the board.

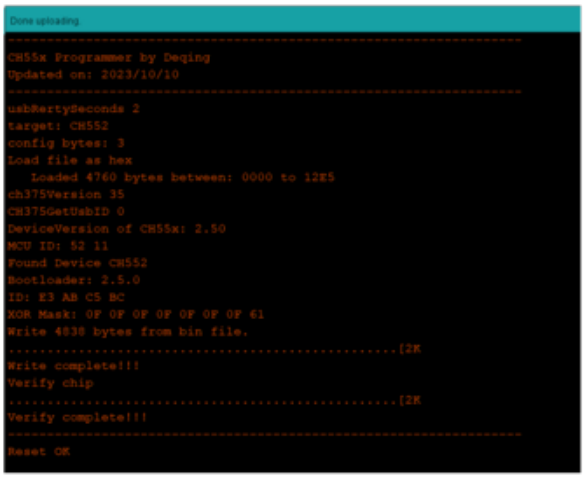


Next, unplug the USB cable from the QT Py. Press and hold the **Boot** button but **do not plug the board back in yet.**



Begin the upload process **without the QT Py plugged in** by clicking the **Upload button** in the Arduino IDE. You'll see the progress at the bottom of the IDE window.

When you see `Compiling libraries...` and `Compiling core...` in the progress output, plug in the QT Py while still holding down the **Boot** button.



If you're successful, you'll see the `Reset OK` message in red at the bottom of the window. If you connect an LED to the **MISO** pin, you should see it blinking.

After this process, you should see your CDC Serial COM port return as a Port option in the Arduino IDE. Using this method you can iterate when working on HID device code without worrying about losing the CDC Serial port.

Linux Troubleshooting Steps

If you are on Linux and find that these instructions don't work for you, try these additional steps. In the terminal enter:

```
cd /etc/udev/rules.d
sudo touch 99.ch55xbl.rules
sudo vi 99.ch55xbl.rules
```

In the rules file, copy and paste the following into the file:

```
# CH55x bootloader
# copy to /etc/udev/rules.d/

SUBSYSTEM=="usb", ATTRS{idVendor}=="4348", ATTRS{idProduct}=="55e0", MODE="0666"
```

Then, save changes and reboot your system. This should allow access to the QT Py.

Windows Troubleshooting Steps

- Windows 10 and 11 may not automatically load a working driver for the CH552 bootloader,
  Installing the CH375 driver (https://adafru.it/1acy) is the recommended option
    - works with ch55xduino
    - works with WCHISPStudio
- Note:
    - Zadigs libusub-win32 also works with ch55xduino, but does not play nice with WCHISPStudio

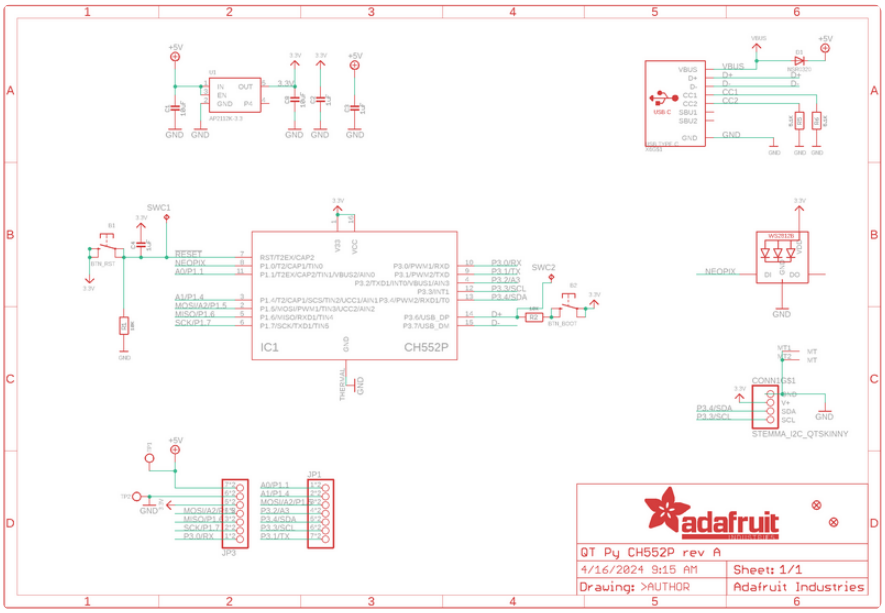
contributed by forum user @rybec full thread (https://adafru.it/1acz)

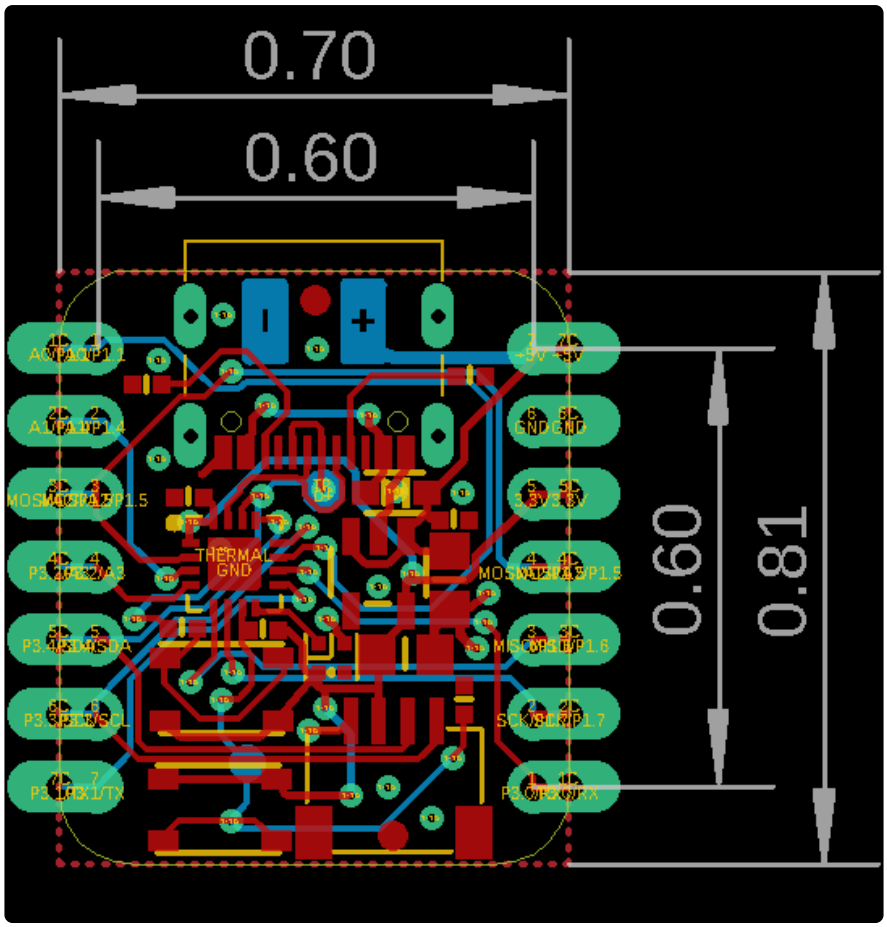
Downloads

Files

- CH552 Product Page (https://adafru.it/19Zf)
- CH552 Datasheet (https://adafru.it/19ZE)
- EagleCAD PCB files on GitHub (https://adafru.it/19ZF)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/1a00)
- PrettyPins pinout PDF on GitHub (https://adafru.it/19ZC)
- PrettyPins pinout SVG on GitHub (https://adafru.it/1a01)

Schematic and Fab Print




0.70
0.60
0.60
0.81
THERMAL
GND