


 **DeqingSun / ch55xduino** Publicforked from [tenbaht/sduino](#)



An Arduino-like programming API for the CH55X

 LGPL-2.1 license 321 stars  182 forks Star Notifications[Code](#) [Issues](#) 7 [Pull requests](#) 2 [Actions](#) [Projects](#) [Security](#) [Insights](#) ch55xduino ▾

Go to file

This branch is [203 commits ahead](#), [7 commits behind](#) tenbaht:development. #139

DeqingSun fix ws2812 include order ...

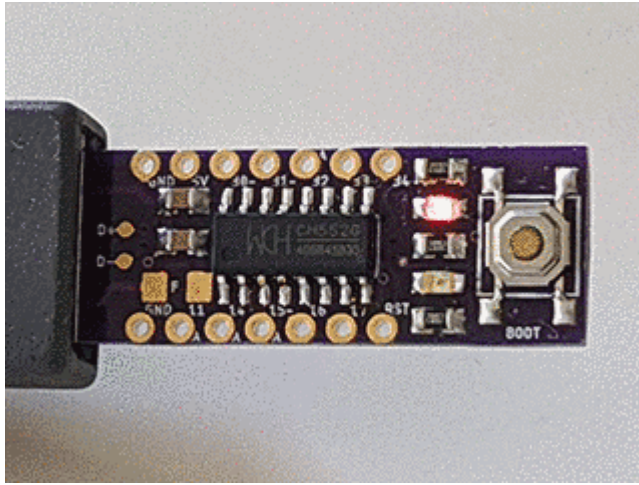
 4 days ago  765[View code](#) README.md

# Ch55xduino: Small Devices Arduino for ch55x devices

Getting started on the Ch55x the easy way. Forked from Sduino project, also based on ch554\_sdcc project

Ch55xduino is an Arduino-like programming API for the CH55X, a family of low-cost MCS51 USB MCU. The project tries to remove the difficulty of setting up a compiling environment. Users can simply write code in Arduino IDE and hit one button to flash the chip to get code running. No configuration or guesswork needed.

CH551, CH552, CH554, may be the lowest part count system that works with Arduino. The minimal system only requires one chip, 2 decoupling capacitors, and one optional pull-up resistor. These features made it ideal for DIY projects.



At this moment the project is still working-in-progress. Support most Arduino functions (Except pulse, shift, tone). Refer to examples in this repo for more info.

## Installation

Automatic IDE integration is supported via the Arduino Boards Manager. This is the recommended way of installation now.

Start the Arduino-IDE. In *File->Preferences, Settings* tab, enter

[https://raw.githubusercontent.com/DeqingSun/ch55xduino/ch55xduino/package\\_ch55xduino\\_mcs51\\_index.json](https://raw.githubusercontent.com/DeqingSun/ch55xduino/ch55xduino/package_ch55xduino_mcs51_index.json)

as an *Additional Boards Manager URL*.

- Open *Tools->Board:...->Boards Manager*
- Find Ch55xduino by typing 'ch' into the search line
- Click on the list entry
- Click on *Install*.

Now you should find a new entry *CH55x Boards* in the list at *Tools->Board:...*

- Choose *CH552 Board* from the list
- open the standard Blink example from *File->Examples->01. Basics->Blink*
- Change pin number in Blink example. For example, if you have LED on P3\_0, you will write pin 30.
- compile it by hitting *Verify*
- If your board is never used with ch55xduino before, you need to make the ch55x chip enter bootloader mode. You need to disconnect USB and unpower ch55x, connect the pull-up resistor on D+ line (generally a 10K resistor between D+ and 5V, controlled by

a push-button or adjacent pads). Then you connect USB. and hit *Upload*. Also, a blank new chip will enter the bootloader automatically.

- If you have used ch55xduino once and your code doesn't crash the USB subsystem, you can simply press *Upload*. Arduino and the firmware will kick the chip into the bootloader automatically.

## Installation without Github access

If you can not access github directly, use this link to a proxy instead.

[https://gh-proxy.deqing.workers.dev/raw.githubusercontent.com/DeqingSun/ch55xduino/playground/mirror/package\\_ch55xduino\\_mcs51\\_proxy\\_index.json](https://gh-proxy.deqing.workers.dev/raw.githubusercontent.com/DeqingSun/ch55xduino/playground/mirror/package_ch55xduino_mcs51_proxy_index.json)

## USB and Serial upload

Ch55xduino supports both USB and Serial upload methods. If the USB port of the CH55x chip is connected to a host computer, the USB method is recommended.

It is also possible to upload via serial on UART1 of CH55x chips (except CH551). To set CH55x into bootloader mode automatically, you can connect RTS or DTR of a serial adaptor to an interrupt pin of CH55x with a capacitor, and add bootloader jumping code in the interrupt handler.

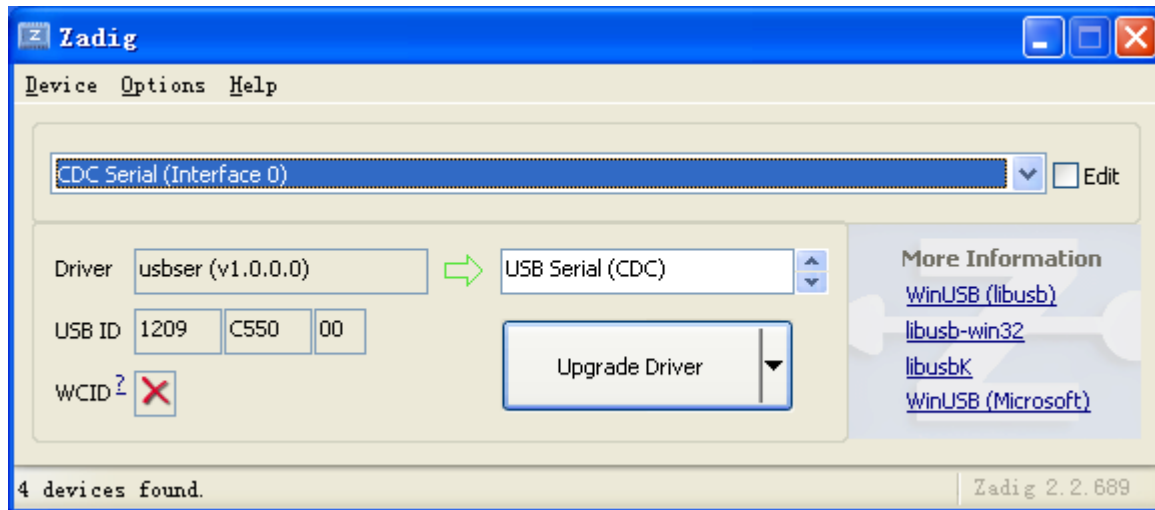
If you want to leave the bootloader, you may send the following bytes at 57600 baud. 57  
AB A2 01 00 01 A4

## Driver for windows

Since 0.0.10, if your Windows automatically installed the driver from wch.cn for the bootloader (4348,55E0), that is fine. The current upload tool can use the default CH375 driver and co-exist with the official [WCHISPTool](#).

If you need to use WinUSB or libusb-win32, the tool will still work.

You can use USB Serial (CDC) driver for the default CDC USB stack (1209,C550). Use [Zadig](#) to install the driver if it did not install automatically.

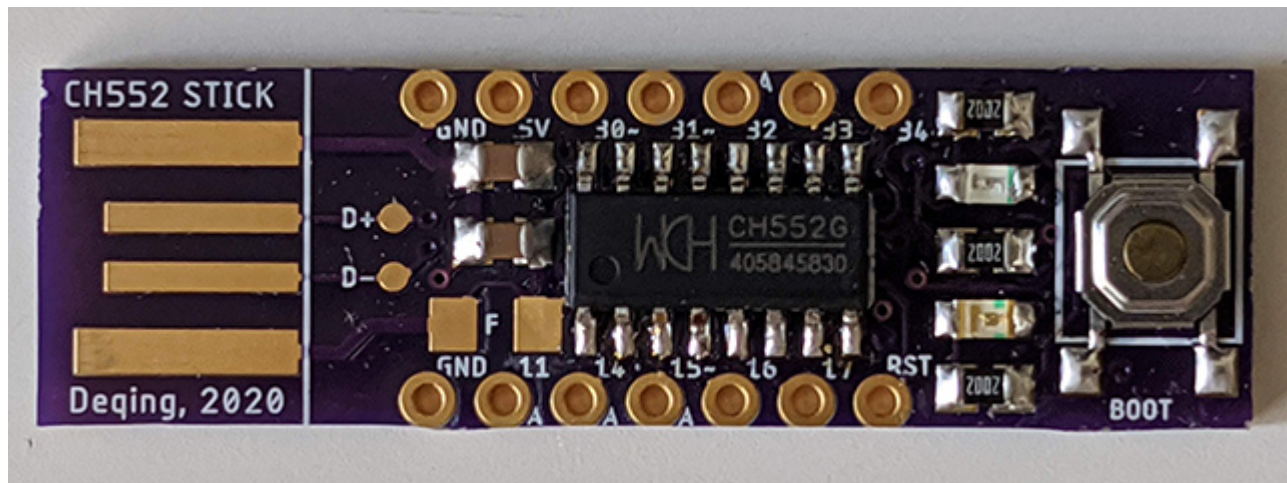


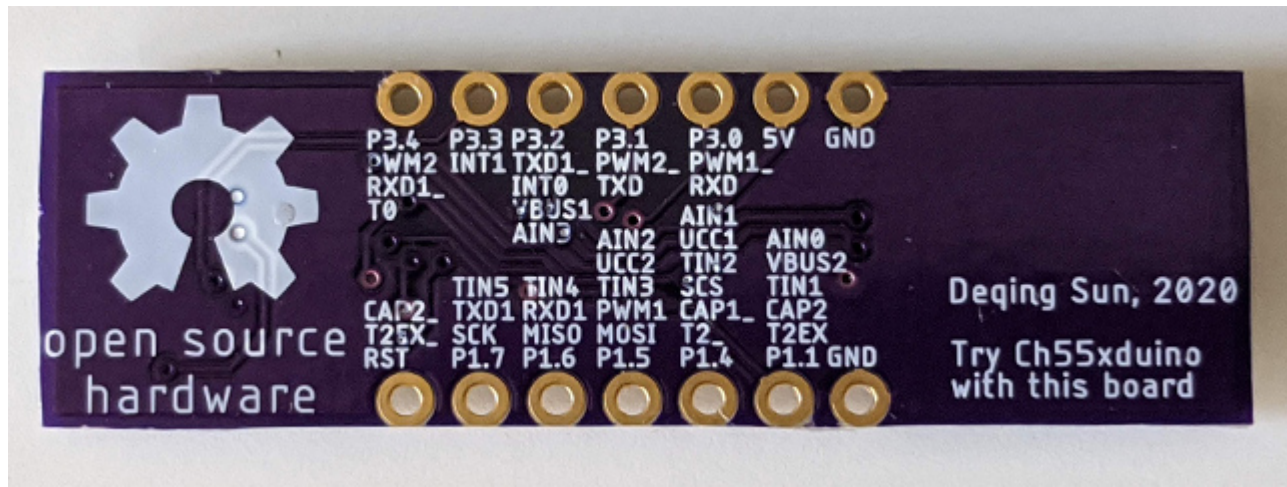
If you tried to emulate another type of USB device without changing the PID/VID, you may need to uninstall the device before installing a new driver.

## Permission for Linux

By default, Linux will not expose enough permission for Arduino to upload the code with the USB bootloader. Copy `99-ch55xb1.rules` in this repo to `/etc/udev/rules.d/` and restart your computer. Otherwise, the upload tool may not find the bootloader device.

## Reference board





There is a small CH552 breakout board design in the "pcb" folder. When fabricated with 1.6mm board thickness, some USB receptacle may be too loose for the PCB. Just add some tape behind the USB connector to increase the thickness.

The button footprint was designed for 6mm buttons, but 5mm one works too.

## Difference to regular Arduino

### Pin names:

Regular Arduino uses continuous numbers to code pins on AVR chips. In my opinion, it makes more sense since AVR uses letters as a port name. But MCS51 core uses numbers as port names. So CH55xduino's pins using the following rule.

$\text{PortNumber} * 10 + \text{PinNumber}$

For example, P1.1 is 11, P3.2 is 32.

### Analog input:

CH552 has an 8-bit, 4 channel analog-to-digital converter on pin P1.1, P1.4, P1.5, and P3.2. So the input range is 0~255, not 1023.

By default, all pins on the MCS51 microcontrollers have internal pull-up resistors enabled. You may need to use `pinMode` to set the pin to `INPUT` to disable the pull-up resistor.

There is no Analog Pin definition such as A0. Just use 11, 14, 15, or 32 for their analog input feature.

### No polymorph functions:

There is no free C++ compiler for MCS51 chip, we can not use polymorph functions. However, since commit 13402 of SDCC, the generic selection is functional. Ch55xduino supports generic selection since 0.0.11.

If you are using a version higher than 0.0.11, the print function can choose a function according to the parameter's type.

For example. If you want to print to the USB-CDC virtual serial port, you can do:

```
USBSerial_print(val);    //val: the value to print - any data type
USBSerial_print(val, format)    //specifies the number base (for integral data
                                types) or number of decimal places (for floating point types)
USBSerial_print(charPointer, length)    //specifies the string length to be
printed
```

It is also possible to do `USBSerial_println`. If you want to print to `Serial0` or `Serial1`, just use `Serial0_print` or `Serial1_print`.

Please note if you pass a character in single quotes, such as `USBSerial_print(' ');`, you will get 44. Because that char got promoted to integer type. You need to either use `USBSerial_print((char) ' ');` or `USBSerial_print(",");`.

They are defined in `genericPrintSelection.h`.

## Memory model:

Unlike most modern architectures including AVR, MCS51 has 2 RAM regions, internal data memory, and external data memory. For CH552, the internal one is only 256 bytes, and the external one is 1024 bytes.

CH55xduino uses the Large Model for SDCC memory models since version 0.0.17. The Large model will allocate all variables in external RAM by default. Variables stored in internal RAM must be declared with the `__data` keyword.

CH55xduino put the stack in internal RAM. So there isn't much space left for variables. If your variable does need fast access, use `__data` when you declare it.

For the default Arduino setting, 148 bytes are reserved for USB endpoints. There will be 876 bytes usable for external RAM.

You can see the memory mapping by opening the map and mem file generated along with the hex file.

## Common Pitfalls when Using SDCC



Note that when some function is called from an interrupt service routine it should be preceded by a `#pragma NOOVERLAY` (if it is not reentrant) . A special note here, `int` (16 bit) and `long` (32 bit) integer division, multiplication & modulus operations are implemented using external support routines developed in ANSI-C, if an interrupt service routine needs to do any of these operations then the support routines (as mentioned in a following section) will have to be recompiled using the `--stack-auto` option and the source file will need to be compiled using the `--int-long-rent` compiler option. [src](#)

With SDCC, interrupt service routine function prototypes must be placed in the file that contains `main ( )` in order for an vector for the interrupt to be placed in the the interrupt vector space. It's acceptable to place the function prototype in a header file as long as the header file is included in the file that contains `main ( )`. SDCC will not generate any warnings or errors if this is not done, but the vector will not be in place so the ISR will not be executed when the interrupt occurs. [src](#)

## Reset Pins

Unlike AVR chips, CH55x will reset when the RST pin is High. The reset pin may be configured as an input pin. But such configuration requires modification of Configuration Information byte, which require an external tool to do so.

## Known issues

---

.

## Included libraries

---

Most parts of the Arduino core system and some Arduino libraries are already ported to C-syntax. The resulting API is still very close to the C++ version and porting an existing application is not hard. Refer to the examples that come with the libraries.

### Communication

- SPI: Real hardware-SPI up to 12MHz.
- SoftI2C: Bit-Bang I2C on any 2 pins.
- WS2812: Bit-Bang WS2812 on any pin. Note some compatible LEDs have different timing and some tweak may be needed.

### Sensors

- TouchKey: Internal 6-channel capacitive touch module wrapper with an adaptive baseline algorithm.

## Compatibility with the Arduino world

---

Since there is no free C++ compiler for the MCS51, it is impossible to do a full 1:1 port of the whole environment as it has been done for the STM32 and the ESP8266.

This is not a drop-in replacement for an AVR, but thanks to some C preprocessor magic the programming API is still very, very similar and it is often enough to just move over the opening bracket of the class instantiation statement and to replace the dot in a method call for an underscore. Check the [migration guide](#) for an overview.


## Supported Systems:

---

Arduino IDE versions 1.8.12 are tested, but most versions  $\geq 1.6.6$  should work.

- Windows: Tested on Windows 7 and XP.
- MacOS: Tested on 10.14.
- Linux: Tested on Ubuntu 20.04 LTS.

### Releases 18

 **0.0.18 release of ch55xduino** Latest  
2 weeks ago

[+ 17 releases](#)

---

### Packages

No packages published

---

### Languages

● C 85.0%   ● C++ 10.8%   ● Shell 1.8%   ● JavaScript 1.3%   ● Python 1.1%