# Mooltipass Bootloader Unlock / Flashing Firmware

You may wish to unlock the bootloader so you can flash custom code. This guide is intended for people who are familiar with embedded firmware, bootloaders, and some coding experience.  Before you begin you will need to email support to get the password for your particular unit.

Email: support[at]themooltipass[dot]com with your serial number.

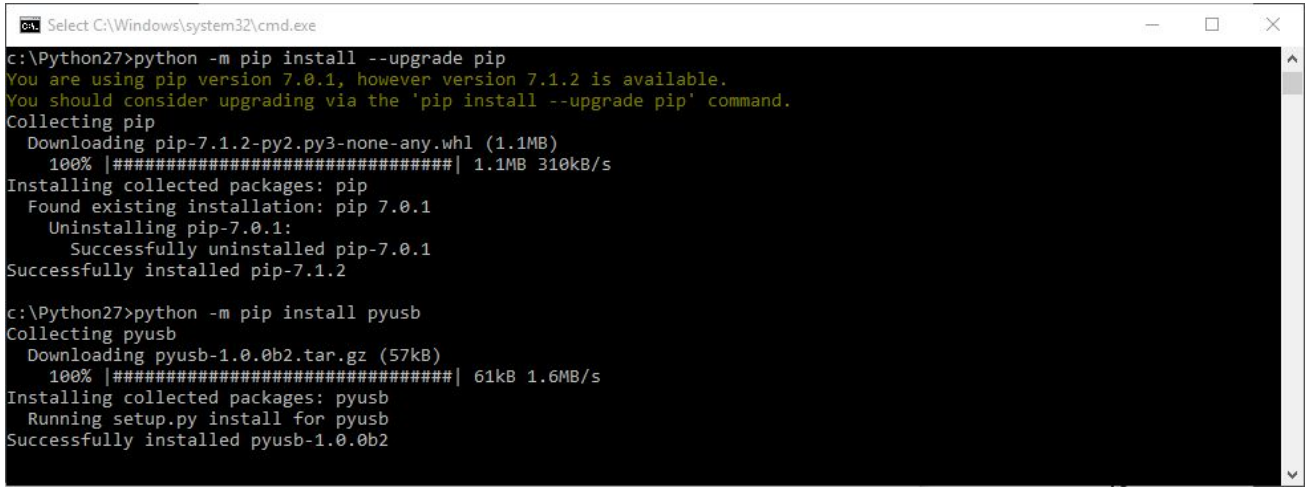
Once unlocked, it will always start in bootloader mode for exactly seven seconds after which if there isn't an update, it will proceed to boot the main firmware as before.

**Note: The first time you do this it will boot with the lights on because the external memory that stores the graphics and credentials will be wiped clean during the unlock.**

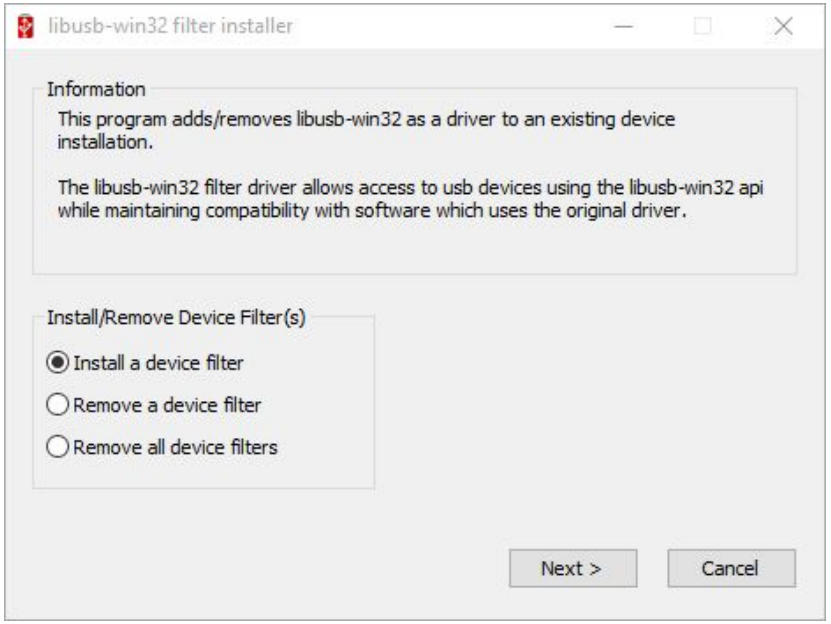
**Be sure to backup your credentials before you start this process.**
To backup use the Chrome App. Under the management tab, click the export button. The exported file has passwords encrypted with AES-256  but usernames are in the clear so be careful with the file. Especially since usernames can be your email address.

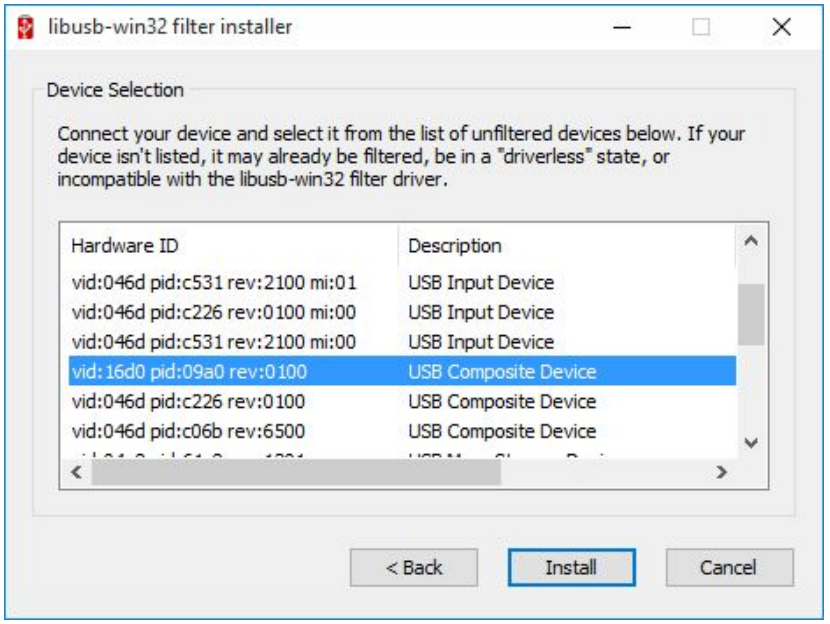
**Prep**

1.) We need the Python Tools that are included in the Github repo. https://github.com/limpkin/mooltipass/tree/master/tools/python_comms
    a.) To download the tools you will need to download the entire repo and extract it. https://github.com/limpkin/mooltipass Click download to the right.
    b.) Extract the contents.
2.) We will need PyUSB and PyCrypto which can be installed via PIP if you have Python installed already.
    a.) https://www.python.org/downloads/ (2.7)
    b.) Once installed you may need to change to that folder if it's not in the path.




c.) You can update PIP if you like
  i.) python -m pip install --upgrade pip
d.) Install pyUSB
  i.) **Windows:** python -m pip install pyusb
  ii.) **Linux:** python -m pip install pyusb --pre
e.) Install pyCrypto
  i.) On Windows you may need Visual C++ Compiler for Python 2.7
    (1) http://aka.ms/vcpython27
  ii.) python -m pip install pycrypto
f.) Windows Only - Libusb
  i.) Extract libUSB from http://sourceforge.net/projects/libusb-win32/files/
  ii.) From the bin folder open folder of your architecture
  iii.) Run *install-filter-win.exe*




iv.)Select the Mooltipass - VID:16D0 PID:09A0

v.) Click Install then Cancel when it's done.

3.) Now we can unlock the bootloader
   a.) Drop to a command prompt in the directory you extracted the Mooltipass repo
   b.) Change directory to ***tools/python_comms***
   c.) Linux: sudo ./mooltipass_coms.py
   d.) Windows mooltipass_coms.py
   e.) Select option 34 and enter password you received via support email
   f.) Accept prompt on Mooltipass

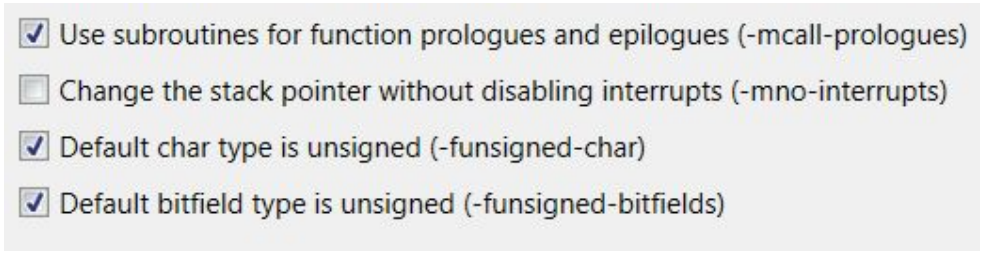
   **Note: You may need to provide full path to Python if it is not in your path. On Windows 10 it will bring up the "which app to use" dialog after a few seconds.**

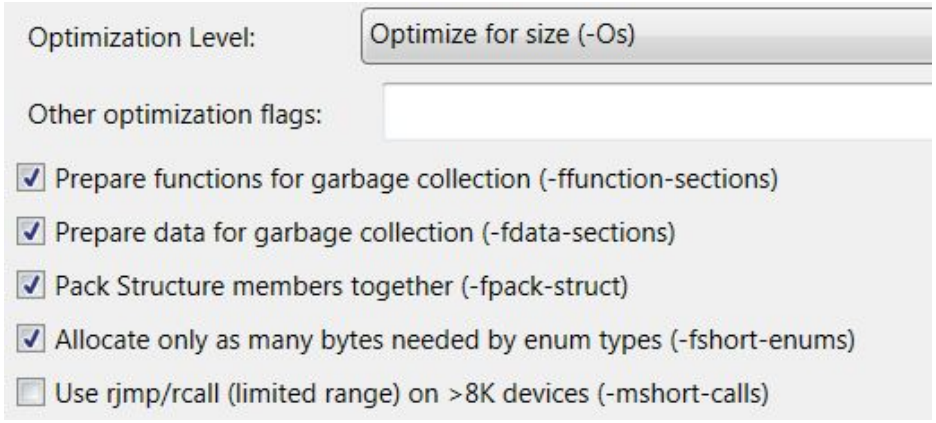
4.) Now you are unlocked and ready to upload code.

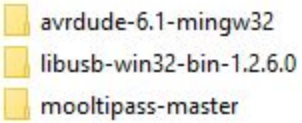
**Flashing Firmware**

Now to the fun part. We have just a few requirements for flashing firmware. If you followed the above section you already have the source code downloaded. Now you just need the following two items.

***Note: Part of this process is with Atmel Studio which is based on MS Visual Studio and therefore only runs on Windows.***

1. **Prep**
    a. AVRDUDE - Download avrdude from http://www.nongnu.org/avrdude/
    b. Atmel Studio - http://www.atmel.com/tools/ATMELSTUDIO.aspx
        i. 6.2 - http://www.atmel.com/tools/STUDIOARCHIVE.aspxhttp://www.atmel.com/tools/STUDIOARCHIVE.aspx
        ii. The source was created with Atmel Studio 6.2. 7 will work but you will need to double check the compiler optimizations settings as they are lost when converting to 7.
        iii. 

        iv. 

    c. Extract AVRDude to a folder next to the source folder. 

2. **Editing Firmware**
    a. Open the project solution in Atmel Studio.
        i. "..\mooltipass-master\source_code\Mooltipass.atsln"
    b. There are few changes to make to setup the defines so you can start working on your own code. In this example we are adding an option to send a keypress when the card is removed.
    c. Firmware Setup
        i. defines.h line 158
            ii. Add the following to create you own Production define. It is really a copy of PRODUCTION_KICKSTARTER_SETUP

```
#elif defined(PRODUCTION_WINLOCK_SETUP)
   #define FLASH_CHIP_4M
   #define HARDWARE_OLIVIER_V1
   #define ENABLE_MOOLTIPASS_CARD_FORMATTING
#endif
```

        iii. defines.h line 73
            1. To enable your define . Comment out the default define and add your own.

```
//#define BETATESTERS_SETUP_PIN
#define PRODUCTION_WINLOCK_SETUP
```

        iv. mooltipass.c line 184
            1. #if !defined(PRODUCTION_SETUP) && !defined(PRODUCTION_KICKSTARTER_SETUP) **&& !defined(PRODUCTION_WINLOCK_SETUP)**

        v. Now any code you add, you can add an “#if defined” around it. For example, this will send a keypress when the card the is removed.
            1. mooltipass.c after line 546

```
#if defined(PRODUCTION_WINLOCK_SETUP)
        usbKeyboardPress(0x0F, 0x08); //GUI+L
#endif
```

3. **Uploading firmware**

a. Build solution by using *F7* or from the Build menu *Build Solution*.

```
Done building target "Build" in project "Mooltipass.cproj".
Done building project "Mooltipass.cproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

b.

c. Once you have worked out your errors you will find the *mooltipass.hex* file here

*\mooltipass-master\source_code\Release*

d. Copy mooltipass.hex to the AVRDude folder. (You could just point directly to it from avrdude but I find it better to make a copy since rebuilding a solution overwrites the file each time.) You may want to keep different versions if you want to go back to a known good build. You could build a hex before you make too many changes so you have a know good hex.

e. Drop to a command prompt inside the AVRDude folder. Set up your parameters at the command line before you reset the Mooltipass. Once reset, you will have seven seconds to run AVRDude to flash your shiny new firmware.

f. Here is the required options for AVRDude - be sure to update the COM port!

***avrdude -v -p atmega32u4 -c avr109 -P com17 -b 57600 -D -U flash:w:mooltipass.hex:i***

**4. Restoring graphics and credentials**

a. The first time you flash the firmware after the bootloader unlock you will get a blank screen since the bitmaps and fonts will have been wiped

b. Following this guide to load the bundle. You can find bundle.img

*..\mooltipass-master\bitmaps\bundle.img*

c. https://docs.google.com/document/d/1e8KFLtd60u1tiUrYj9k-2n8DgEtU1MwChWBIX1U0P7E

d. If you have problems getting the Chrome App to “see” the Mooltipass go into Chrome extensions setup and disable/enable the Chrome App.

e. You can now restore your credentials from backup using the Chrome App.

**5. Bootloader Locking - {//Section In Progress}**

a. Once you have the Mooltipass how you want it you can either leave it as is or relock the bootloader. This way you don’t have to wait 7 seconds every time you power it up.

**6. Final Notes**

a. Once you reload the graphics and credentials you won’t need to restore them again each time you flash a new firmware.