

[illegible]

PU22+ -	DATA / SCEX output	=	DIGITAL PIN 4
	SUBQ DATA	=	DIGITAL PIN 8
	SUBQ CLOCK	=	DIGITAL PIN 9
	WFCK / GATE	=	DIGITAL PIN 10
Mainboard <= PU20	DATA / SCEX output	=	DIGITAL PIN 4
	SUBQ DATA	=	DIGITAL PIN 8
	SUBQ CLOCK	=	DIGITAL PIN 9
	WFCK / GATE	=	DIGITAL PIN 3 (Best to just tie to ground on the PS1 mainboard, but you can use this pin on the MCU)

How to prep the MCU!

Credit where it's due:

UberNee is *somewhat* a port of PSNEE Version 7 by Rama, it's being re-written from scratch in my own style and I've made my own algorithm for the hysteresis. I also use a different method of modulation for PU22+. The MCU does not 'sample' the WFCF frequency and instead creates it's own, hence, wiring will differ. The advantages of course are the price of these LGT8F328P MCU's, they're dirty cheap. There may be future updates for unlocking of PAL PSOne consoles (these have an additional region lockouts) and JP consoles (also region locked).

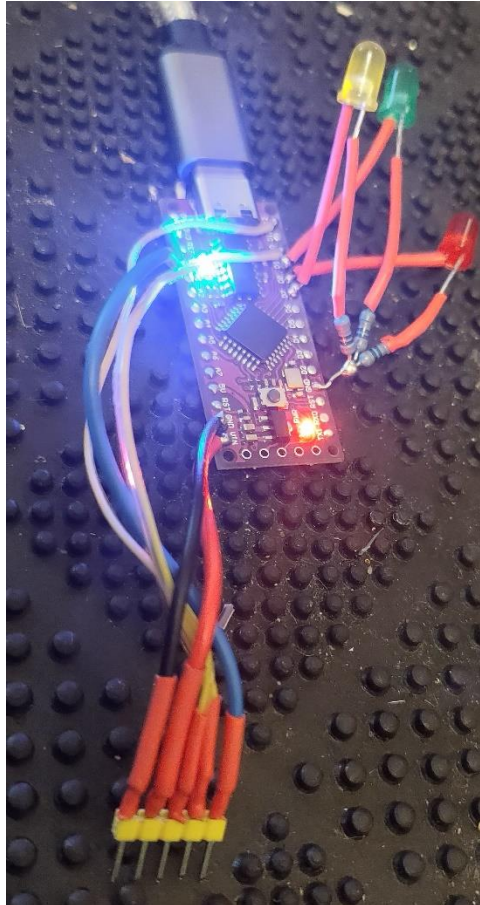
Basically, the bytes it captures from the Subq data line were already discovered - It just needed to be re-written for the MCU. Also credit to 'zoinknoise', he's a massive douche bag who attacks me constantly on Discord, BUT ... he found a handful of games a few years ago that PSNEE couldn't launch as they contained a different combination of bytes around a certain part of the TOC area (basically where the DRM exists on authentic discs) which are indicative of the laser being in the area it finds the region strings, I call these magic keys, they're also known as 'symbols'.

The Modchip only sends these magic keys when it knows the laser head is in the correct area of the disc, hence – ***it's 100% stealth.***

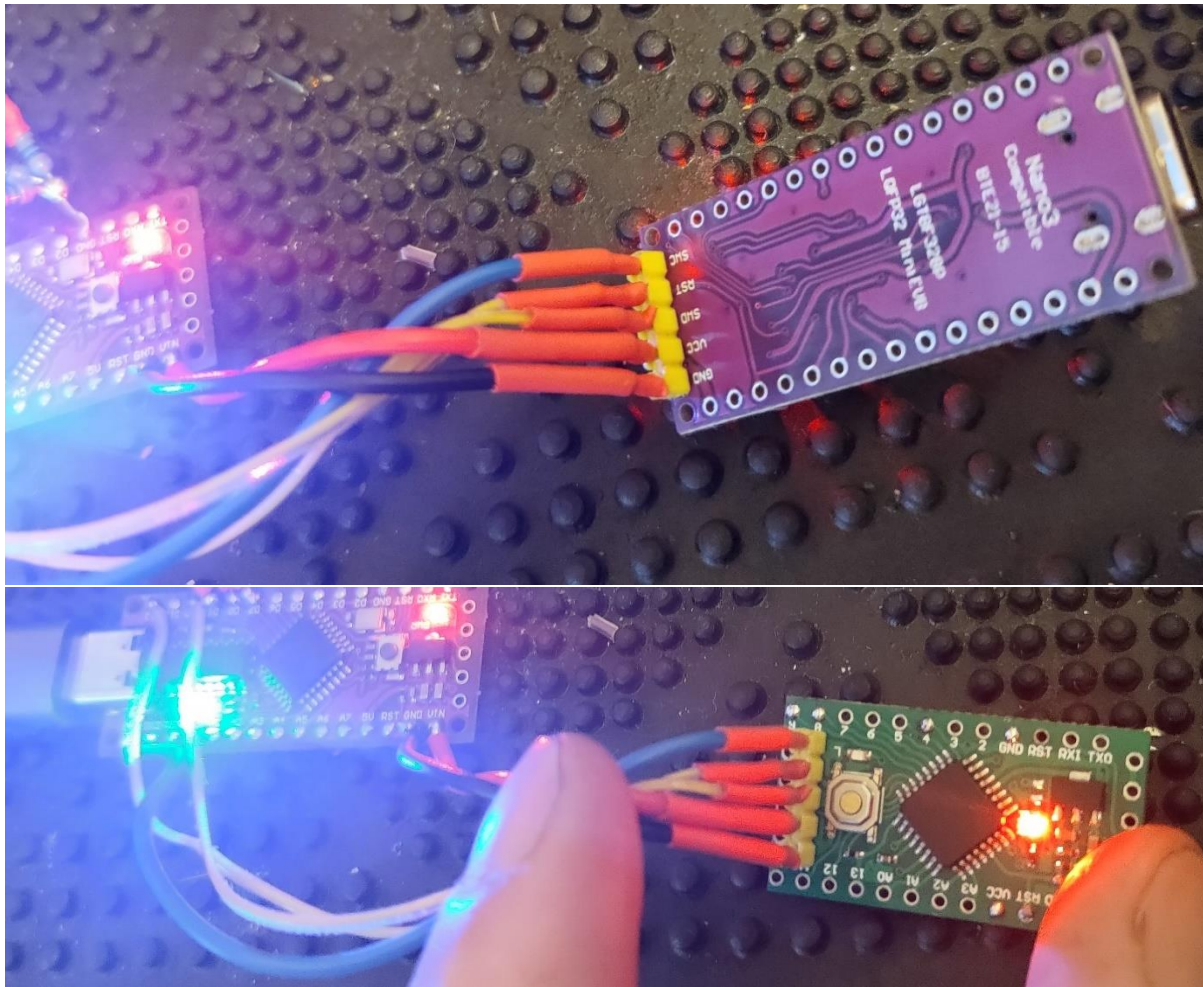
Further credit: brother-yan for the LGTISP flashing tool
LaZsolt for the timing accurate library that doesn't cause crashes or bootloops.

Hardware

You'll simply need a spare LGT8F328P to flash UberNee onto another LGT8F328P via ISP.



ISP Flasher	UberNee
D13	SWC
D12	SWD
D10	RST
GND	GND
VIN	VCC



It's important to note that the Micro version pins are back to front, hence, one will plug in upside down and the other facing up as shown. I've used red and black for GND and VCC so I can align the pins correctly. If you plug these in the wrong way – you'll see smoke.

The LED's probably aren't required, but strangely enough, I honestly had no luck with this until uncommenting the LED lines and hooking them up. I spent countless hours trying to use an arduino UNO to flash the LGT8F328P MCU via ISP at first. Don't go that route. I've provided the same sketches I used, with the same minor alterations.

These being the uncommenting of LED lines and addition of

```
#define SERIAL_RX_BUFFER_SIZE 250:
```

You also need to use the correct library for UberNee as other guides will tell you to use different (and incompatible) libraries for this procedure. We will also be using 'avrdudeess v7' which differs from other ISP flashing guides.

***** Another important note is that the larger variant with the USB port needs the 3.xxV rail from the PS1 mainboard wired to the 3.3v pin and NOT THE VIN pin. The smaller version though, goes to VIN. *****

Software Stuff

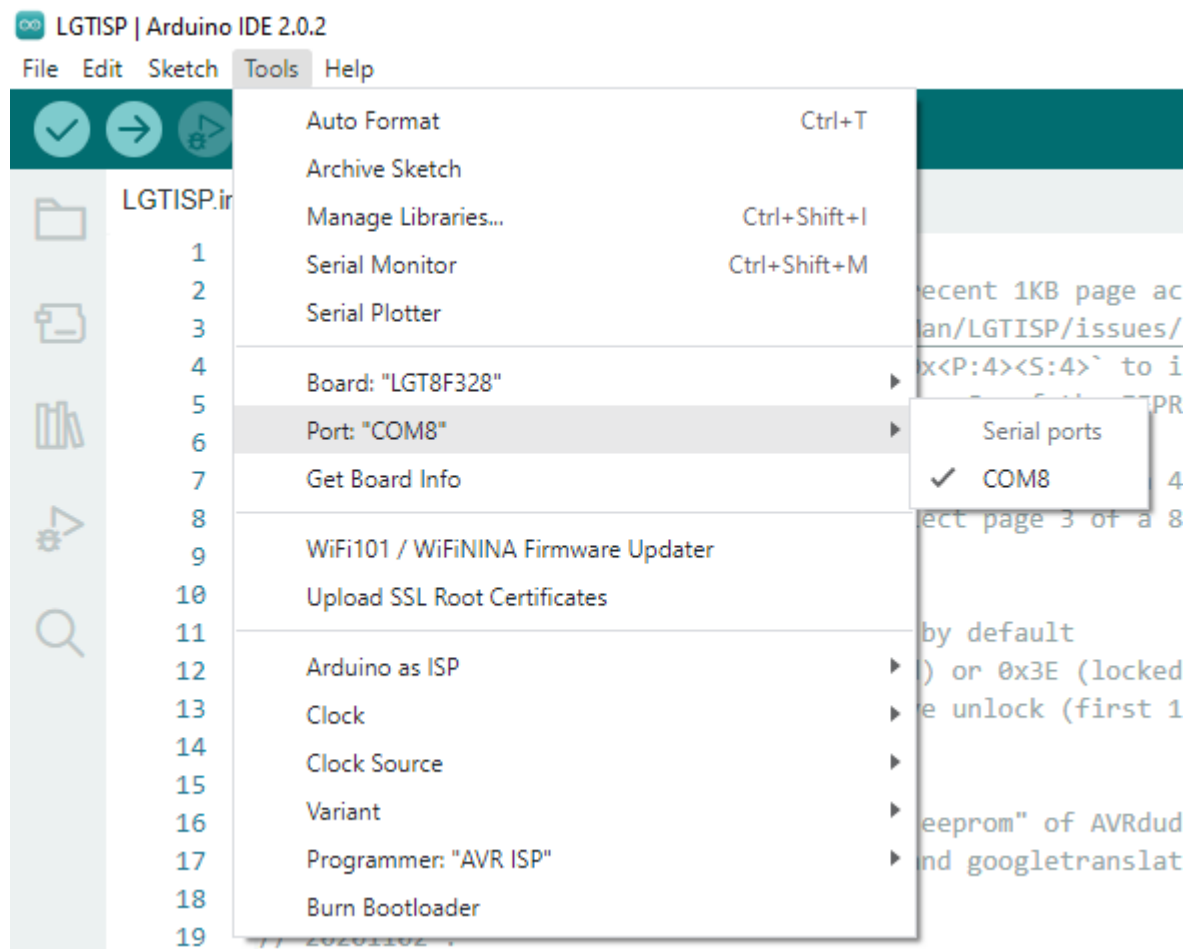
First and foremost, the library in use is '[lgt8f-1.0.7-alpha.1.zip](#)'

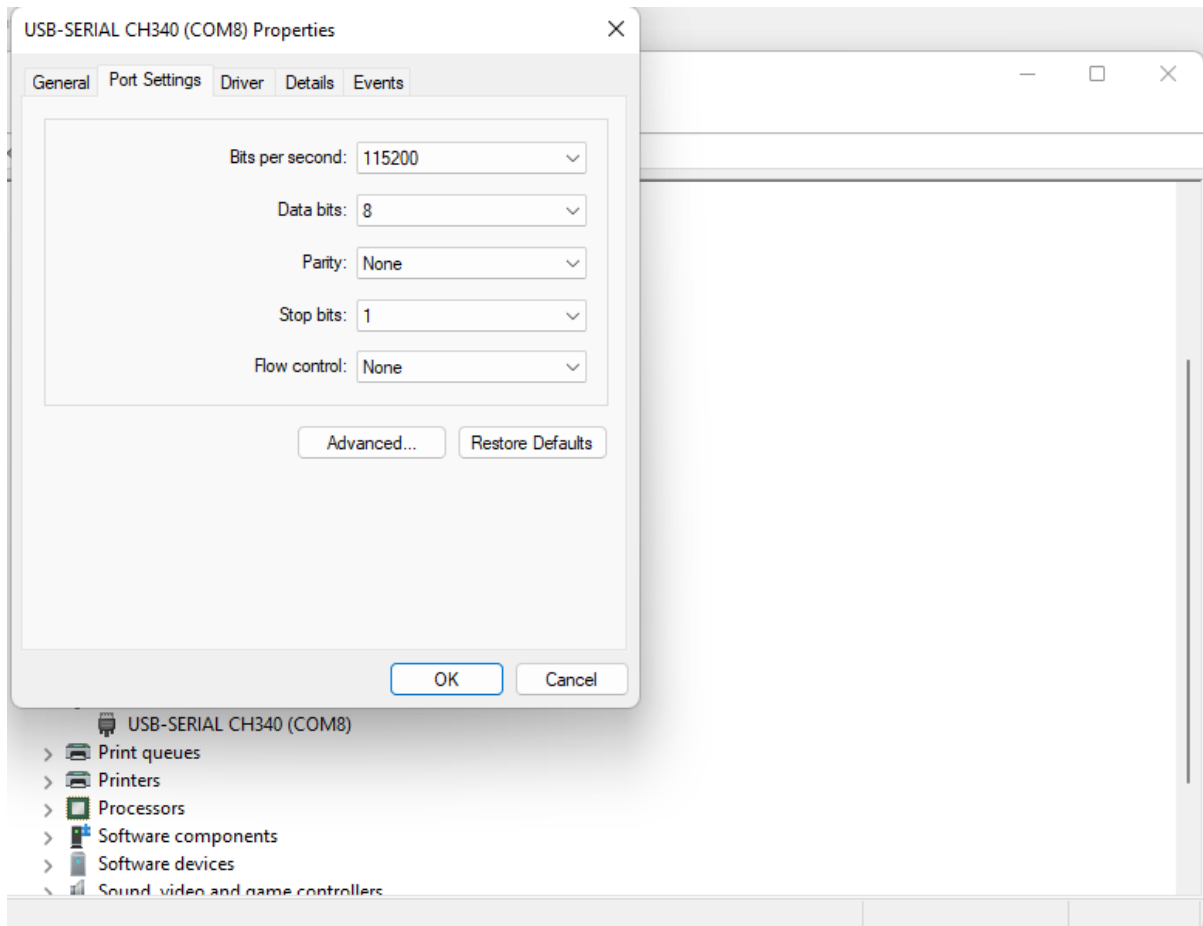
The files and guide are in the Uberchip repository.

Credit to [LaZsolt](#) for this timing accurate library that doesn't cause crashes and bootloops.

Make sure this is installed before proceeding.

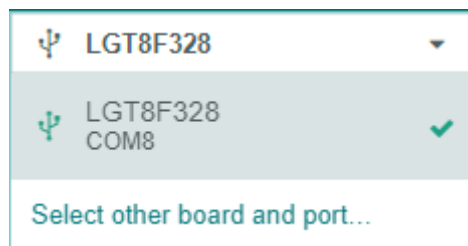
Now we can open up our 'LGTISP.ino' into the arduino IDE – make sure the correct port is selected for your LGT8F328P that you are going to turn into an ISP flasher (plug it in).





Before flashing, just to be thorough, go into device manager and find your USB->Serial device (COM # will be the same as the port in arduino) and set it to 115200, and hit 'OK'.

We also need to make sure we've selected the LGT8F328P board, leave the other settings on default.



Use '**CTRL+U**' for upload or the 'sketch' drop down menu then 'upload' back in the arduino IDE, you'll see a successful flash down in the output window. If you've got any issues so far, you'll have to do some googling and see where you've gone wrong. It's a little out of scope of this guide.

That's our ISP flasher created! You can now plug the board you are turning into UberNee into your new flasher as shown in the pictures on Page 3. You may need to put a little diagonal pressure on the headers to make proper contact when we do the flash later.

UberNee sketch to HEX conversion

Open the UberNee ino into Arduino IDE, you can close the other ISP sketch

```
//***** MAGIC KEYS *****/
// EUROPE / AUSTRALIA / NEWZEALAND/ UK = SCEE
// USA = SCEA
// JP = SCEI
//*****

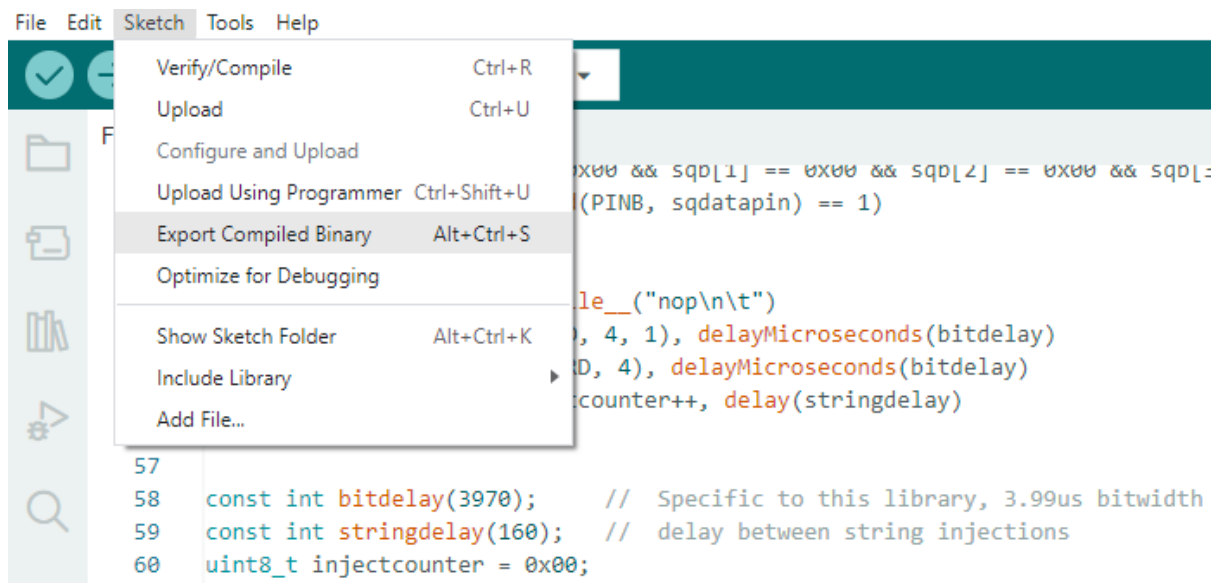
//***** Debug Mode *****/
//      yes / no ?
const bool DEBUG_MODE = no           //<-----
//*****

#define SELECT_MAGICKEY SCEE         //<-----

int TWEAK_DRIVE = 5;                //<-----
```

Before converting the sketch to a hex for ISP flashing, make your choices and selections here. It won't hurt to leave debug mode on, but is pointless, unless you are wanting to do some Dev stuff. When it's on, you can leave the power and ground wire off the installation on the PS1 mainboard and have it powered by your PC's USB port to monitor the Chip in action over the serial monitor – it will show you the packets it's capturing off the disc along with the hysteresis value.

It can be used to make adjustments. Once you want to use it as a stand-alone chip in the console, you need to wire ground and power.



Now that we've made our selections, simply use the 'Sketch' drop down menu and 'Export Compiled Binary', or '**ALT+CTRL+S**'.

We'll now have the files we need as shown below (circled).

Documents > Arduino > FUckman > build > LGT8fx Boards.avr.328

Name	Status	Date modified	Type	Size
FUckman.ino.eep	✖	10/12/2022 10:18 PM	EEP File	1 KB
FUckman.ino.elf	✖	10/12/2022 10:18 PM	ELF File	30 KB
FUckman.ino.hex	✖	10/12/2022 10:18 PM	HEX File	10 KB
FUckman.ino.with_bootloader	✖	10/12/2022 10:18 PM	Virtual CloneDrive	30 KB
FUckman.ino.with_bootloader.hex	✖	10/12/2022 10:18 PM	HEX File	3 KB

Don't worry about the stupid name. At some stage I was frustrated and branched the source code off to a new file with major alterations and just forgot to change it. This is the hex file we wan't with no bootloader 😊

NOW CLOSE ALL THE ARDUINO IDE STUFF, we don't want anything getting in the way of our ports for the next step.

AVRDUDESS

Open up this badboy, leave everything on default unless specified to change below.

Programmer: Atmel AVR ISP.

Port (-P): The port the ISP flasher you created is plugged into.


Flash: The UberNee.hex (bootloaderless, prev. Page)

EEPROM: The *.eep file (prev. Page)

Baud rate (-b): 115200

MCU (-p): LGT8F328P

Options: Tick the “erase flash and eeprom (-e)”

 AVRDUDESS 2.14 (avrdude version 7.0)

Programmer (-c)		
Atmel AVR ISP		
Port (-P)	Baud rate (-b)	Bit clock (-B)
COM8	115200	
Flash		
C:\Users\fsan\OneDrive\Documents\Arduino\Fuckman\build\LGT8fx Boards.avr.328\Fuckman.ino.hex		
<input checked="" type="radio"/> Write	<input type="radio"/> Read	<input type="radio"/> Verify
<input type="button" value="Go"/>		
EEPROM		
C:\Users\fsan\OneDrive\Documents\Arduino\Fuckman\build\LGT8fx Boards.avr.328\Fuckman.ino.eep		
<input checked="" type="radio"/> Write	<input type="radio"/> Read	<input type="radio"/> Verify
<input type="button" value="Go"/>		
Options		
<input type="checkbox"/> Force (-F)	<input checked="" type="checkbox"/> Erase flash and EEPROM (-e)	
<input type="checkbox"/> Disable verify (-V)	<input type="checkbox"/> Do not write (-n)	
<input type="checkbox"/> Disable flash erase (-D)	Verbosity <input type="text" value="0"/>	
<input type="button" value="Program!"/> <input type="button" value="Stop"/>		

MCU (p)

LGT8F328P

Flash: 32 KB

EEPROM: 1 KB

Detect

Presets

Default

Manager

Fuses & lock bits

L

H ☐ Set fuses

E [Fuse settings](#)

LB

☐ Set lock

Bit selector

Additional command line args

Hit program!

As mentioned earlier, you may need to apply some pressure on the header pins (unless you directly soldered and planned on desoldering after).

```
-c avrisp -p lgt8f328p -P COM8 -b 115200 -e -U flash:w:"C:\Users\lfsan\OneDrive\Documents\Arduino\Fuckman\build\LGT8fx Boards.avr.328\Fuckman.ino.

>>>: avrdude
Fuckman.ino.hex: 3,550 / 32,768 Bytes (10.83%)
Fuckman.ino.eep: 0 / 1,024 Bytes (0.00%)
~ ~ ~ ~ ~
>>>: avrdude -c avrisp -p lgt8f328p -P COM8 -b 115200 -e -U flash:w:"C:\Users\lfsan\OneDrive\Documents\Arduino\Fuckman\build\LGT8fx Boards.avr.328\Fuckman.ino.

avrdude.exe: AVR device initialized and ready to accept instructions

Reading | ##### | 100% 0.01s

avrdude.exe: Device signature = 0x1e950f (probably lgt8f328p)
avrdude.exe: erasing chip
avrdude.exe: reading input file "C:\Users\lfsan\OneDrive\Documents\Arduino\Fuckman\build\LGT8fx Boards.avr.328\Fuckman.ino.hex"
avrdude.exe: input file C:\Users\lfsan\OneDrive\Documents\Arduino\Fuckman\build\LGT8fx Boards.avr.328\Fuckman.ino.hex auto detected as Intel Hex
avrdude.exe: writing flash (3550 bytes):

Writing | ##### | 100% 0.88s

avrdude.exe: 3550 bytes of flash written
avrdude.exe: verifying flash memory against C:\Users\lfsan\OneDrive\Documents\Arduino\Fuckman\build\LGT8fx Boards.avr.328\Fuckman.ino.hex:
avrdude.exe: input file C:\Users\lfsan\OneDrive\Documents\Arduino\Fuckman\build\LGT8fx Boards.avr.328\Fuckman.ino.hex auto detected as Intel Hex

Reading | ##### | 100% 0.44s

avrdude.exe: 3550 bytes of flash verified
avrdude.exe: reading input file "C:\Users\lfsan\OneDrive\Documents\Arduino\Fuckman\build\LGT8fx Boards.avr.328\Fuckman.ino.eep"
avrdude.exe: input file C:\Users\lfsan\OneDrive\Documents\Arduino\Fuckman\build\LGT8fx Boards.avr.328\Fuckman.ino.eep auto detected as Intel Hex
avrdude.exe: writing eeprom (0 bytes):

Writing | ##### | 100% 0.00s

avrdude.exe: 0 bytes of eeprom written
avrdude.exe: verifying eeprom memory against C:\Users\lfsan\OneDrive\Documents\Arduino\Fuckman\build\LGT8fx Boards.avr.328\Fuckman.ino.eep:
avrdude.exe: input file C:\Users\lfsan\OneDrive\Documents\Arduino\Fuckman\build\LGT8fx Boards.avr.328\Fuckman.ino.eep auto detected as Intel Hex

Reading | ##### | 100% 0.00s

avrdude.exe: 0 bytes of eeprom verified

avrdude.exe done. Thank you.
```

We're done, you can install the chip.