# Central station build & programming Instructions

## Overview

The following must be programmed for the central station to work:

* FTDI USB to serial convertor
* ATMEGA2560, this needs to be programmed to behave like an Arduino Mega, this is achieved by loading the Arduino bootloader and associated fuses.
* ESP8266-01 module, this is programmed as an access point, using software developed by Phillip Gahtow, as default it ships with AT modem commands

Version 1 hardware requires some changes:

* The analog inputs are incorrectly wired up, net Av needs to be disconnected form PC1 and connected to PK2 (pin 87), and the Current sense net (G1) needs to disconnected and connected to PK1 (pin 88).
* Components on the H Bridge need to be removed, D3 is replaced by a 0R 0603 resistor and 0R resistor fitted across the base and emitter of Q8 the 2 resistors and D1 are not fitted

## Programming FTDI 232RL

Instructions to install the FT\_Prog utility can be found at:

<https://www.ftdichip.com/Support/Documents/AppNotes/AN_124_User_Guide_For_FT_PROG.pdf>

Ensure that the Jumper SJ2 is set for 5V IO (towards the external programmer socket)

* Connect the USB port to the PC, and ensure open the FT\_Prog utility
* Scan for connected devices, if the device is not found, check the PCB for assembly errors.
* If required change the manufacturer & description

## Hardware Setup

Plug the AVRISP mkII USB connection into a spare USB port on the PC. Plug the AVRISP ribbon cable connector into the 3 by 2 header on the Arduino board. Make sure that you connect it the right way around. Pin 1 is marked on the AVRISP connector and on the Arduino header.

Power up the Arduino either from the PC via a USB cable or by an external power supply.

## Burning the Arduino Bootloader Mega2560

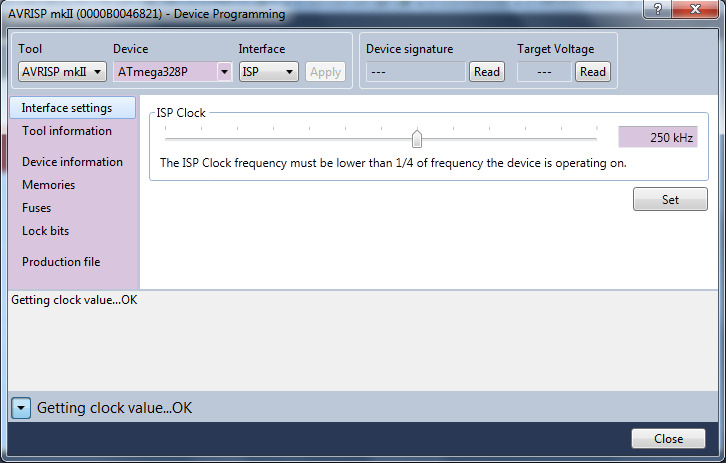
The bootloader is included with the Arduino IDE software. In version 1.0.1 of the Arduino software, you will find the bootloader in the Arduino folder:

**C:\Program Files (x86)\Arduino\hardware\arduino\avr\bootloaders\stk500v2**

The bootloader in this folder that must be loaded to the Arduino Mega is **stk500boot\_v2\_mega2560.hex**

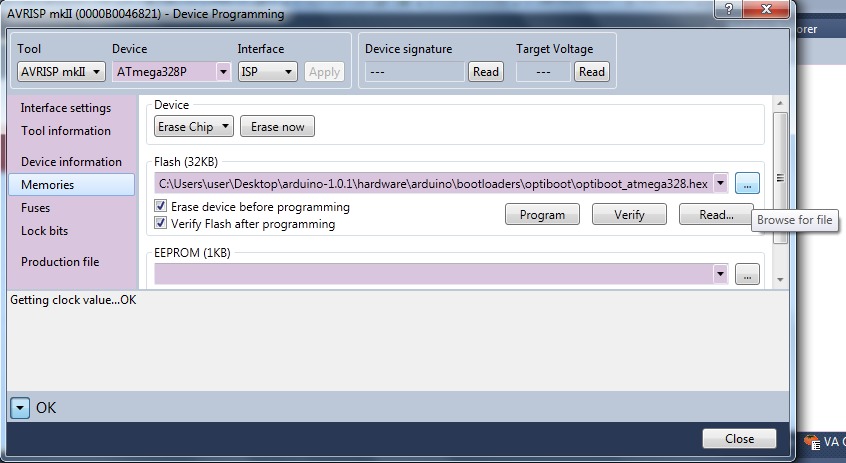
## Steps to Burn the Bootloader

1. Connect the hardware as described above
2. Start Atmel Studio
3. In Atmel Studio, click **Tools → Device Programming**
4. In the Device Programming dialog box, select **AVRISP mkII** in the Tool drop-down box, **ATmega2560** in the Device drop-down box and **ISP** in the Interface drop-down box

[](https://startingelectronics.org/tutorials/arduino/arduino-uno-burning-bootloader-avr-studio/atmel-studio-settings.jpg)

**Atmel Studio Interface Settings - click for a bigger image**

1. Click the Apply button
2. You can click the **Read** button under Device signature to make sure that you can connect to the ATMEGA2560 microcontroller
3. Click Memories in the left pane of the dialog box
4. Browse to the bootloader hex file using the **...** button and then select the hex file
5. Click the **Program** button

[](https://startingelectronics.org/tutorials/arduino/arduino-uno-burning-bootloader-avr-studio/burning-the-bootloader.jpg)

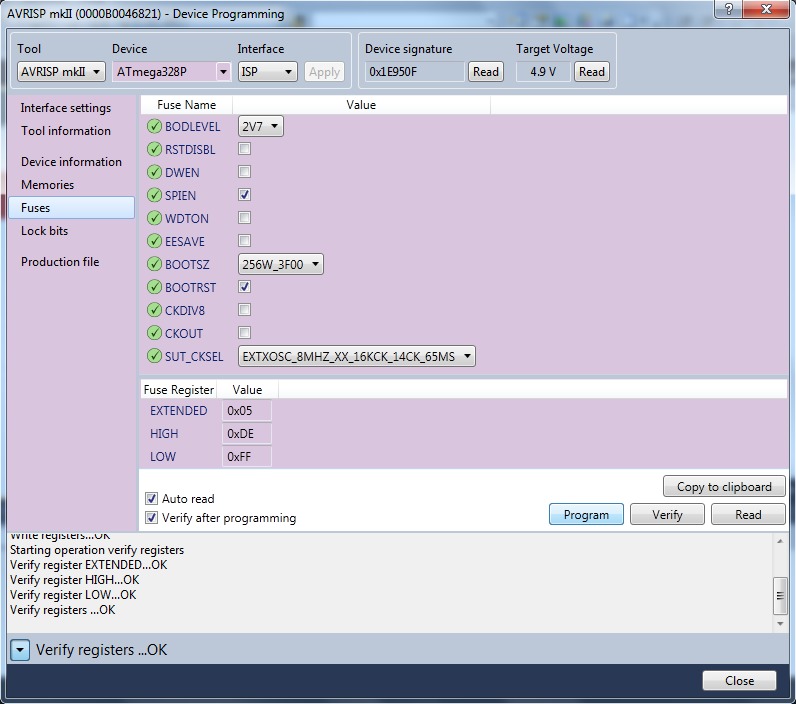
**Burning the Arduino Uno Bootloader - click for a bigger image**

1. After programming has finished, click Fuses in the left pane

Change the fuse values to:  
mega2560.bootloader.extended\_fuses=0xFD

mega2560.bootloader.high\_fuses=0xD8

mega2560.bootloader.low\_fuses=0xFF  
  
Click the **Program** button

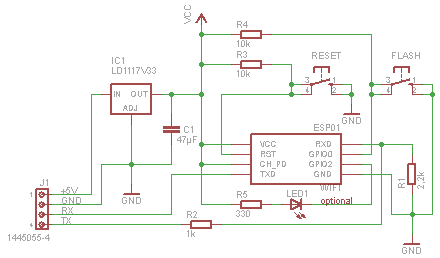
[](https://startingelectronics.org/tutorials/arduino/arduino-uno-burning-bootloader-avr-studio/arduino-uno-fuse-settings.jpg)

**Arduino Uno Fuse Setting - click for a bigger image**

1. You can now close Atmel Studio
2. Start the Arduino IDE and load a sketch to test that the bootloader is working, e.g. load the Blink sketch to flash the on-board LED

## ESP Access point setup

For that you need to connect a serial to USB adapter shown below, the 3.3V FTDI adapter outputs a 5V supply but communicates at 3.3V levels, hence it requires a regulator.

  
Program the sketch on Phillip’s website.

<http://pgahtow.de/wiki/index.php?title=Zentrale/en#Z21_WLAN_Add-On_Sketch>  
Install the ESP library to your Arduino IDE. <http://pgahtow.de/wiki/index.php?title=Arduino#ESP8266>  
  
To upload the sketch to the ESP you need to have button attached to start the ESP into the bootloader.  
<http://pgahtow.de/wiki/index.php?title=Datei:ESP8266-01_Modul.png>  
You need to press Reset and Flash at the same time, then start the upload from the Arduino IDE. Release reset and load down the Flash button during the download, when the download completes release the Flash button.  
  
This you need to do only once. If the sketch runs inside the ESP it builds a Access Point and a WiFi Client at the same time. You can configure the WiFi if you connect to the Access-Point with the default password 12345678 on the IP <http://192.168.0.111>. It will open a website where you can specify the AP settings and WiFi Client connection setting. Both connections can work at the same time.

## Burning the Arduino Bootloader Sanguino

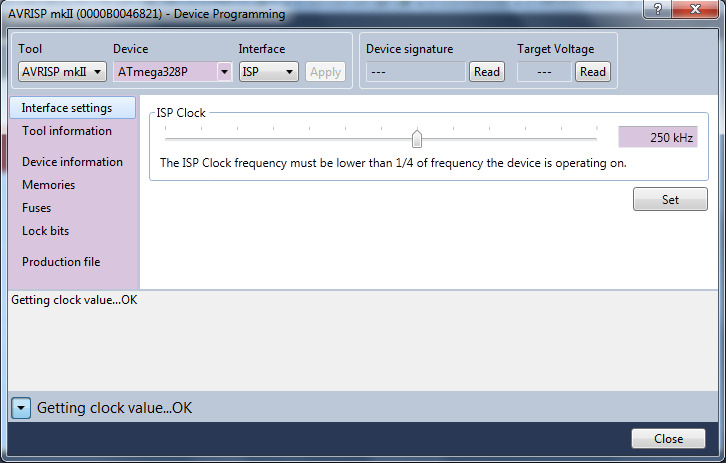
The bootloader is included with the Arduino IDE software. In version 1.0.1 of the Arduino software, you will find the bootloader in the Arduino folder:

**C:\Program Files (x86)\Arduino\hardware\Sanguino\bootloaders\optiboot**

The bootloader in this folder that must be loaded to the Arduino Mega1284 is **optiboot\_atmega1284p.hex.hex**

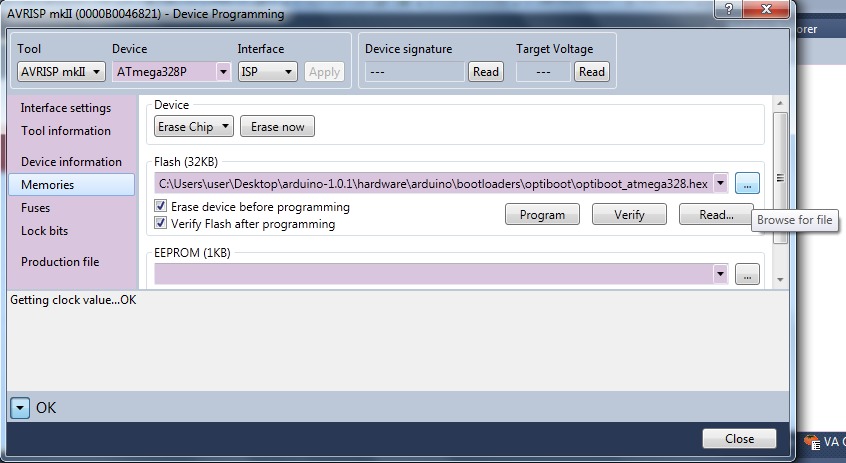
## Steps to Burn the Bootloader

1. Connect the hardware as described above
2. Start Atmel Studio
3. In Atmel Studio, click **Tools → Device Programming**
4. In the Device Programming dialog box, select **AVRISP mkII** in the Tool drop-down box, **ATmega1284p** in the Device drop-down box and **ISP** in the Interface drop-down box

[](https://startingelectronics.org/tutorials/arduino/arduino-uno-burning-bootloader-avr-studio/atmel-studio-settings.jpg)

**Atmel Studio Interface Settings - click for a bigger image**

1. Click the Apply button
2. You can click the **Read** button under Device signature to make sure that you can connect to the ATMEGA1284p microcontroller
3. Click Memories in the left pane of the dialog box
4. Browse to the bootloader hex file using the **...** button and then select the hex file
5. Click the **Program** button

[](https://startingelectronics.org/tutorials/arduino/arduino-uno-burning-bootloader-avr-studio/burning-the-bootloader.jpg)

**Burning the Arduino Uno Bootloader - click for a bigger image**

1. After programming has finished, click Fuses in the left pane

Change the fuse values to:

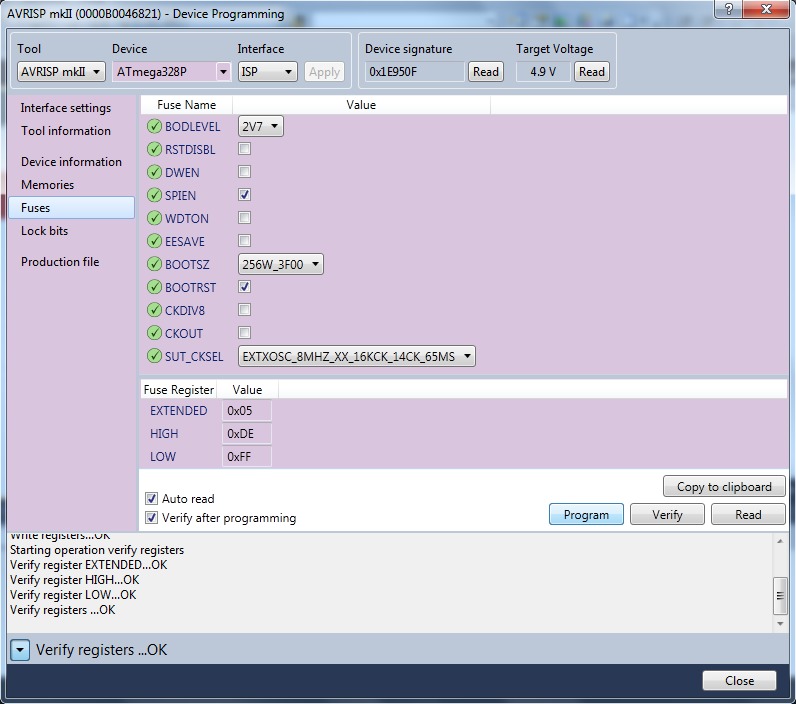
atmega1284p.bootloader.extended\_fuses=0xFD

atmega1284p.bootloader.high\_fuses=0xDE  
atmega1284p.bootloader.low\_fuses=0xFF

atmega1284p.bootloader.unlock\_bits=0x3F

atmega1284p.bootloader.lock\_bits=0x0F

1. Click the **Program** button

[](https://startingelectronics.org/tutorials/arduino/arduino-uno-burning-bootloader-avr-studio/arduino-uno-fuse-settings.jpg)

**Arduino Uno Fuse Setting - click for a bigger image**

1. You can now close Atmel Studio
2. Start the Arduino IDE and load a sketch to test that the bootloader is working, e.g. load the Blink sketch to flash the on-board LED.

## Arduino Settings

Set the board type as Mighty Core 1284, pinout as Bobino.

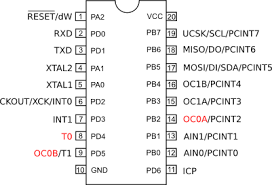
## Programming Open DCC Booster

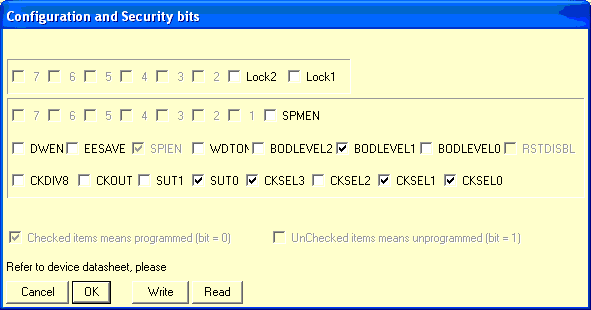
Use the AVR Dragon

Select Tiny 2313

Program memory from

C:\Users\GRedpath\Documents\DCC\Tiny Booster 2\Booster\_Timer.hex



Set default fuses in Atmel Studio.

## Booster wiring

X3-1 (nearest the heatsink) is 0V in from PSU

|  |  |
| --- | --- |
| Pin | Function |
| X3-1 | 0 Volts In |
| X3-2 | +15V In |
| X2-1 | DCC1-OUT |
| X2-2 | DCC2-OUT |
| X1-1 | DCC1-IN |
| X1-2 | DCC2-IN |

