Guava Bootloading Documentation

Written By: Team Guava

Last Updated: 11/14/2020

Supplies:

- Arduino Uno
- 1x Micro USB cable
- 1x USB Type A/B cable
- Bare arduino
    - 2x Capacitors (*22 pF*)
    - Crystal Clock (*8 MHz*)
    - 1x Resistor (*100* Ω)
    - 1x Push button
    - Microcontroller of Choice (*ATMEGA1284*)
    - Misc wires

Software:

- Arduino IDE
- Microcontroller library (Mighty)

Getting Started:

In order to program your microcontroller, you will need to burn the bootloader into it. You can use the Arduino Uno as an ISP (In-System Programmer) to burn bootloaders. First, you will want to construct your bare arduino with your microcontroller of choice (Guava uses the *ATMEGA1284*) or find a working PCB. You can follow the instructions on the Arduino website to construct your bare arduino through this link or through the documentation on the wiki.

Configuring the Arduino Uno:

Before you begin bootloading, you will need to connect the microcontroller to the Arduino Uno. You will be using the SPI (Serial Peripheral Interface) protocol as compared to I2C protocol. Their differences can be found through this link. For the purposes of this tutorial, we will be using the *ATMEGA1284*, but it will work for any microcontroller with SPI capabilities.. A schematic for the bootloading configuration can be found in Figure 1 below:

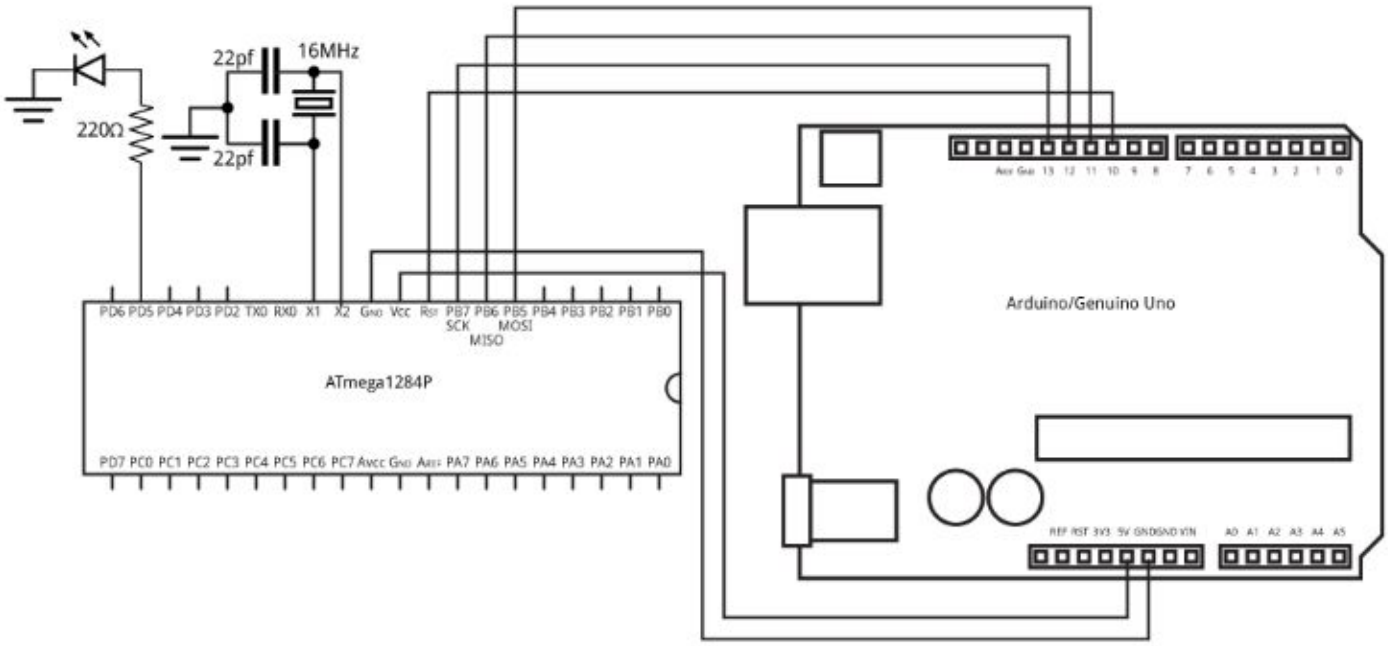

Figure 1: Connecting an Arduino Uno to an ATMega1284 for ISP programming

You may be using a different microcontroller than the ATMega1284, in which the pins will be very different. In that case, you will want to follow the following table for connecting the Uno to your bare arduino:

Microcontroller Pin	Arduino Uno Pin
RST	10
MOSI	11
MISO	12
SCK	13
VIN	5V or 3.3V
GND	GND

Figure 2: Pin configurations for connecting a microcontroller to the Arduino Uno for ISP Programming

Burning the Bootloader:

Now that the Uno is configured for ISP programming, you can open up the Arduino IDE software. Make sure that the appropriate core is installed for your microcontroller. You can find the cores from this github repository or you can install them through this tutorial. Once it is installed properly, you can check to see if it is installed by checking under Tools → Boards. If you see a submenu labelled MightyCore then you have installed it correctly. While the Uno is also plugged into the computer, also verify that the port is recognized under Tools → Ports. The Arduino Uno should appear here while it is plugged into the computer.

There are two main steps to bootloading using the Arduino Uno. In the first step, we have to upload the ArduinoISP file to the Uno. The ArduinoISP file can be found under File → Examples → 11.ArduinoISP. Next plug in the Arduino Uno to the computer and select the “Arduino / Genuino Uno” under Tools → Boards. Change the port to match the one that the Uno is connected to under Tools → Ports. We will be using the “AVR mkII” programmer for this first part, which can be found under Tools → Programmer. Now you can upload the ArduinoISP code to the Uno. If it does not upload correctly, check the ports and board selection. A screenshot of the correct configuration can be found below in Figure 3:

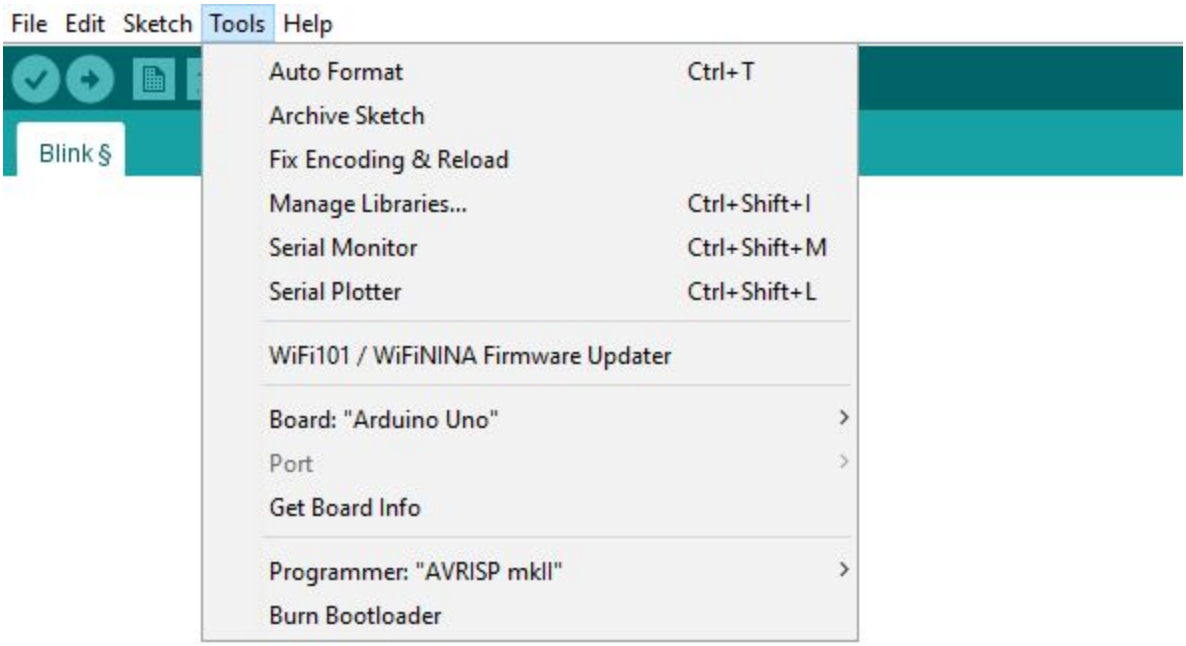

Figure 3: Settings for uploading Arduino.ISP to the Arduino Uno

In the next part, we will be burning the bootloader onto the microcontroller. We will be using the same port from the first part, but you will need to change the board to the microcontroller that you are using under Tools → Boards. The programmer will also need to be set to "Arduino as ISP". Now you can burn the bootloader by going under Tools and selecting "Burn Bootloader". After this step, your microcontroller should be bootloaded properly. You can test it by trying to upload another sketch through an FTDI or by pressing the reset button and checking if the LED flashes.

TLDR:

1. Build bare arduino
2. Connect Arduino Uno to bare arduino using table in Figure 2
3. Install appropriate microcontroller core
4. Open ArduinoISP file under File → Examples → 11.ArduinoISP
5. Under Tools make sure the following settings are selected:
    a. Boards → “Arduino / Genuino Uno”
    b. Port → “COM# (Arduino / Genuino Uno)”
    c. Programmer → “AVR mkII”
6. Upload code
7. Change the settings under Tools again to the following:
    a. Boards → “(microcontroller core here)”
    b. Port → “COM# (Arduino / Genuino Uno)”
    c. Programmer → “Arduino as ISP”
8. Burn the bootloader under Tools → Burn Bootloader

References:

Bootloading tutorial:

- http://www.technoblogy.com/show?19OV#bootloader

I2C and SPI Differences:

- https://aticleworld.com/difference-between-i2c-and-spi/

MightyCore Github:

- https://github.com/MCUdude/MightyCore

MightyCore Installation Tutorial:

- https://elementztechblog.wordpress.com/2016/10/28/mightycore-an-arduino-core-for-the-atmega16-atmega32-atmega324-and-more/

Standard, Bobuino, and Sanguino Pinouts:

- https://github.com/MCUdude/MightyCore#pinout

Further Readings:

ATMEGA328P Bootloader:

- https://www.circuito.io/blog/atmega328p-bootloader/
- https://www.arduino.cc/en/Tutorial/BuiltInExamples/ArduinoToBreadboard

ISP's:

- https://www.quora.com/What-is-in-system-programming
- https://en.wikipedia.org/wiki/In-system_programming

Communication Protocols (I2C, SPI, UART):

- https://www.seeedstudio.com/blog/2019/09/25/uart-vs-i2c-vs-spi-communication-protocols-and-uses/
- https://www.rfwireless-world.com/Terminology/UART-vs-SPI-vs-I2C.html