Guava Bootloading Documentation

Written By: Team Guava

Last Updated: 10/29/2020

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. Fortunately, we 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*).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:

Once you have finished setting up your bare arduino, you'll want to connect the Arduino Uno so that it can be configured to program as an ISP. We 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:

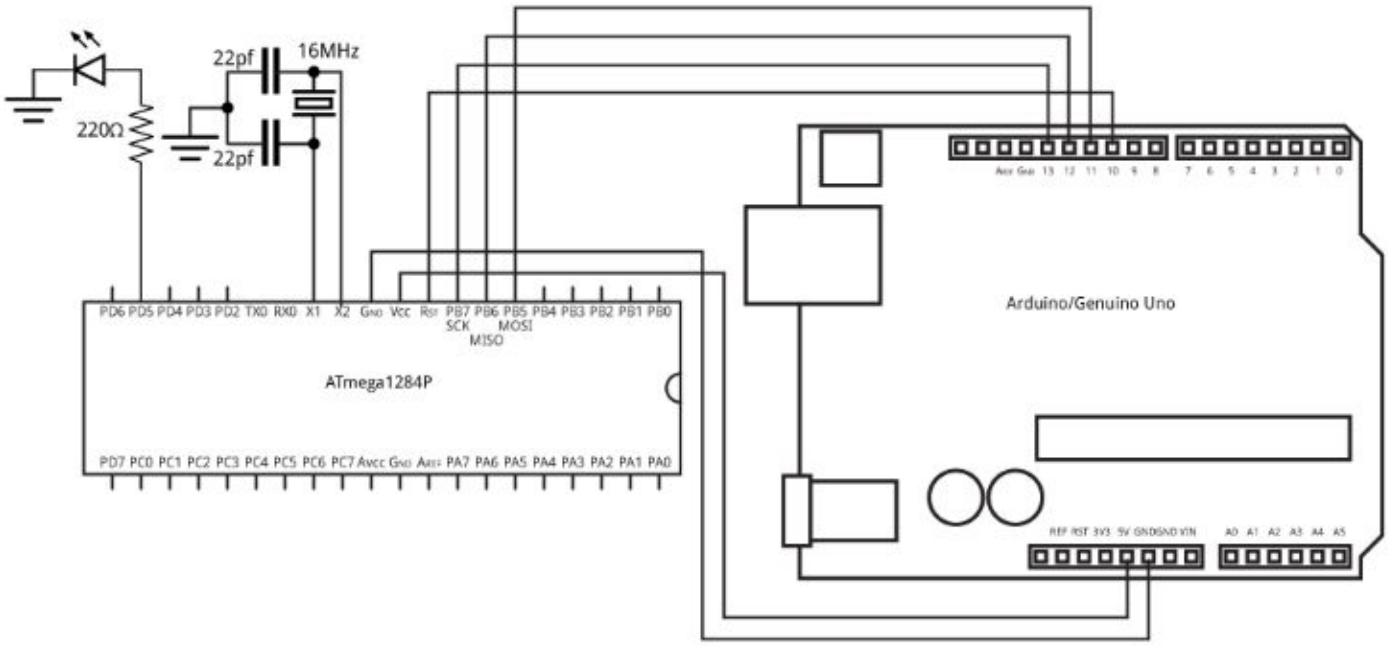

Figure 1: Connecting an Arduino Uno to an ATMega1284 for ISP programming

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

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. You will want to open the ArduinoISP file which can be found under File → Examples → 11.ArduinoISP. Make sure that the appropriate core is installed for your microcontroller. You can find the cores from this github repository or you can install it 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.

Before we can burn the bootloader, make sure that the Uno is plugged into the computer with Arduino IDE running. There are two main parts to configuring the Uno. In this first part, we have to upload the ArduinoISP file to the Uno. To do this, 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.

In this next part, we will be burning the bootloader onto the microcontroller. Change the board to the microcontroller that you are using under Tools → Boards. We will be using the same port from the first part. Change the programmer to “Arduino as ISP”. Now we can burn the bootloader by going under Tools and selecting “Burn Bootloader”. After this step, the Uno should begin burning the bootloader onto your

microcontroller. You can test if your board is bootloaded properly by trying to upload new code through an FTDI or by pressing the reset button and checking if the LED flashes. By default, every bootloader should have a reset functionality built into it, and will usually be port 0 or 1.

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:

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