

Program ATtiny85 Using Uno

You can program an ATtiny85 microcontroller using an Arduino Uno by **turning the Uno into an In-System Programmer (ISP)**. This process requires a few simple hardware connections and a one-time software setup in the Arduino IDE.

Required Materials

- Arduino Uno (or Nano/Mega)
- ATtiny85 microcontroller
- Breadboard
- Jumper wires (6 recommended)
- ~~10µF electrolytic capacitor~~
- Tested With Arduino IDE version 2.3.7
- Optional: LED and 470-ohm resistor for testing

Step 1: Prepare the Arduino IDE

You need to add support for ATtiny boards to the Arduino IDE.

1. Open the Arduino IDE and go to **File > Preferences**.
2. In the **Additional Boards Manager URLs** field, paste the following URL:
https://raw.githubusercontent.com/damellis/attiny/ide-1.6.x-boards-manager/package_damellis_attiny_index.json

If you have other URLs, separate them with a comma.

3. Click **OK**.
4. Go to **Tools > Board > Boards Manager**.
5. Search for "attiny" and install the board package ("attiny" by David A. Mellis).

6. Restart the Arduino IDE.

Step 2: Configure the Arduino Uno as an ISP

You will upload a special sketch to the Uno to make it act as a programmer.

1. Connect the Arduino Uno to your computer via USB.
2. In the IDE, go to **Tools > Board** and select your Arduino Uno model.
3. Go to **File > Examples > 11. ArduinoISP** and open the sketch.
4. Upload the sketch to the Uno.

Step 3: Wire the ATtiny85 to the Arduino Uno

Disconnect the Uno from the computer first. Now, wire the two microcontrollers together on a breadboard using the Serial Peripheral Interface (SPI) pins.

Arduino Uno Pin	ATtiny85 Pin (Physical Pin No.)
-----------------	---------------------------------

+5V	VCC (Pin 8)
------------	--------------------

GND	GND (Pin 4)
------------	--------------------

Pin 10	RESET (Pin 1)
---------------	----------------------

Pin 11 (MOSI)	Pin 5 (Pin 5)
----------------------	----------------------

Pin 12 (MISO)	Pin 6 (Pin 6)
----------------------	----------------------

Pin 13 (SCK)	Pin 7 (Pin 7)
---------------------	----------------------

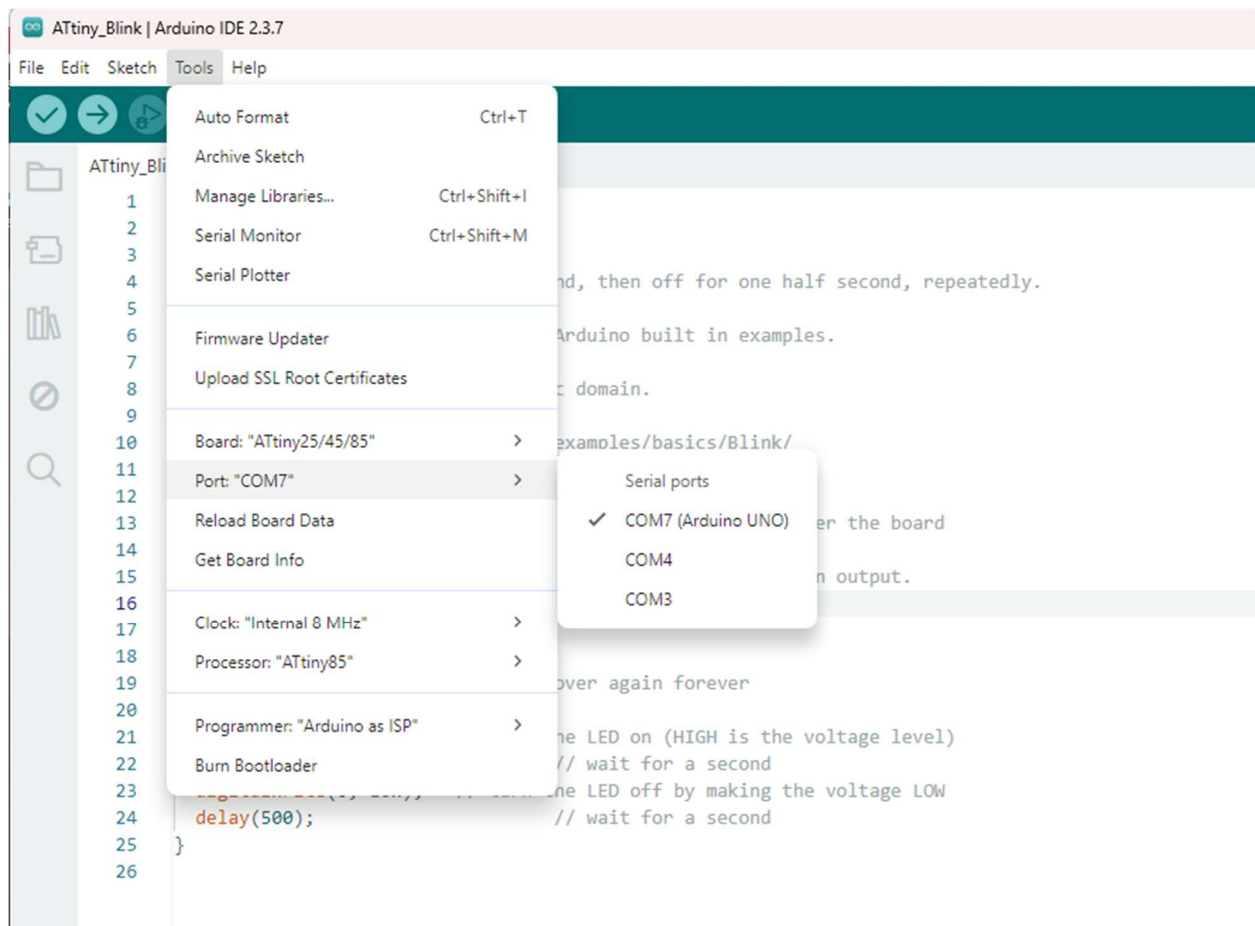
Important: Place a ~~10 μ F electrolytic capacitor~~ between the ~~RESET~~ pin and ~~GND~~ on the ~~Arduino Uno~~. This prevents the Uno from auto-resetting, allowing it to act as an ISP for the ATtiny85. Ensure the capacitor's negative leg connects to GND.

Step 4: Burn the Bootloader to the ATtiny85

Burning the bootloader sets the fuses on a new ATtiny85 chip and configures it to use its internal clock speed. This only needs to be done once per chip (unless you need to change the clock settings).

1. Reconnect the Arduino Uno to your computer.
2. In the IDE, go to **Tools > Board** and select the **ATtiny25/45/85**
3. Go to **Tools > Processor** and select **ATtiny85**.
4. Go to **Tools > Clock** and select **Internal 8 MHz**
5. Go to **Tools > Programmer** and select **Arduino as ISP**.
6. NOTE: keep the COM port of your Uno that is the programmer.
7. Go to **Tools > Burn Bootloader**. You should see "Done burning bootloader" in the console.

See figure below:



Step 5: Upload Your Sketch

Now you can upload your custom code to the ATtiny85.

1. Keep all wiring the same as in Step 3.
2. Open an example sketch, like **Examples > 01.Basics > Blink**.
3. Change the pin number from to , as Pin 0 on the ATtiny85 corresponds to physical Pin 5.
4. Ensure the correct board settings (ATtiny85, 8 MHz internal) are still selected.
5. Instead of clicking the standard "Upload" button, go to **Sketch > Upload Using Programmer**.
6. Once uploaded, you can disconnect the ATtiny85 from the Uno and power it independently. For the Blink sketch, above, connect 5 volt power to ATtiny85 pin 7; GND to ATtiny85 pin 4, connect a 470 ohm resistor to ATtiny pin 5. Connect the

other end of the resistor to the anode (longer pin) of an LED. Connect the cathode (shorter pin) of the LED to GND. When power is applied, the LED should blink.

NOTE: from this point on, you can change the program in the Arduino IDE. Compile the new program and re-execute only step 5 after verifying the programmer wiring and settings. You do not need to burn the bootloader again.

Mapping ATtiny85 Physical Pins to Arduino IDE Pins

ATtiny85 Pin (Physical)	Function	Arduino IDE Digital I/O Pin
Pin 1	Reset	-
Pin 2	PB3 (Analog 3)	3
Pin 3	PB4 (Analog 2)	4
Pin 4	GND	GND
Pin 5	PB0 (MOSI/PWM)	0
Pin 6	PB1 (MISO/PWM)	1
Pin 7	PB2 (SCK/Analog 1)	2
Pin 8	VCC (2.7V - 5.5V)	5V

Note: The ATtiny85 PB5 (Physical Pin 1) is typically used for Reset and cannot be easily used for I/O.