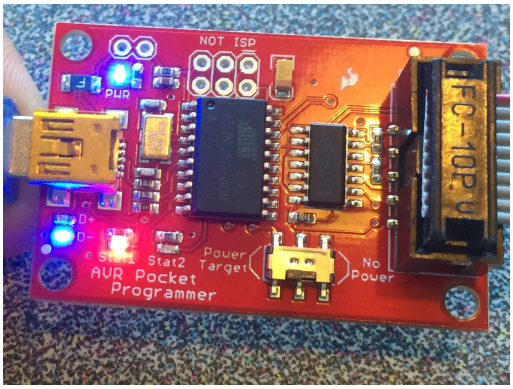
(Windows Only)

Atmel Studio Tutorial

Hardware Prerequisites

Programmer Board - USBTinyISP

I usually use the “AVR Pocket Programmer” They run about 15$ on amazon. Other AVR programmers can also be used.



AVR Target Chip

There are some restrictions as to which chips you can program but most (Atmel AVR chips) are supported by Avrdude
(For this tutorial I am using the Attiny85)

Downloads

Atmel Studio

Atmel Studio is an IDE, Compiler, and Project manager all-in-one.
It provides a programming environment and helps write and convert C, C++, or Arduino into code that can be uploaded to the AVR microcontrollers.

Download “Atmel Studio 7” You should be able to google the newest version.
Or download here: http://www.atmel.com/microsite/atmel-studio/

AVRDUDE+GCC

Avrdude is the program that uploads your compiled code onto the avr Microcontrollers.

Download “WinAVR” You should be able to google the newest version.
Or download here: https://sourceforge.net/projects/winavr/

USBTinyISP Drivers

Get the newest drivers so that you can upload without issue

Google the drivers for the USBtinyISP.
Or download here: https://learn.adafruit.com/usbtinyisp/drivers

Setup

Avrdude

First we need to make sure that Avrdude is works

Go to Command Console (type Cmd into your search bar to find it)

Type:
Avrdude

Hit Enter.

A long list of options should pop up.

If you successfully Installed the WinAVR package avrdude should be a recognized executable command.

Possible Errors

You may have to restart your computer after installing WinAVR in order for avrdude to be recognized.

UsbTinyISP

Now that we have Avrdude working let make sure that our programming board works.

To check that that your computer can communicate with the programming board (In my case the AVR Pocket Programmer) we need to use Avrdude to send it a command.

In Cmd type:
Avrdude -c usbtiny -p attiny85

> /*
> If you are not using the attiny85 enter
> **Avrdude -c usbtiny**
> A list of other compatible microcontrollers will pop-up.
> In place of **attiny85** use the name of your chip
> */

Hit Enter.

```
C:\>avrdude -c usbtiny -p attiny85

avrdude: initialization failed, rc=-1
         Double check connections and try again, or use -F to override
         this check.


avrdude done.  Thank you.
```

You should receive this error message.
This is because we haven't hooked the programmer to any chip. However this means that we have established a connection between the computer and the programmer.

Possible Errors

Make sure that you have installed the most recent drivers for your programmer.

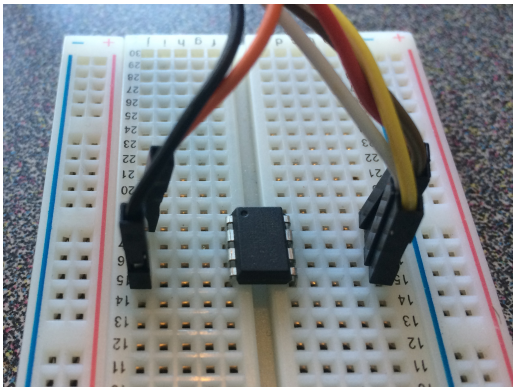
Microcontroller

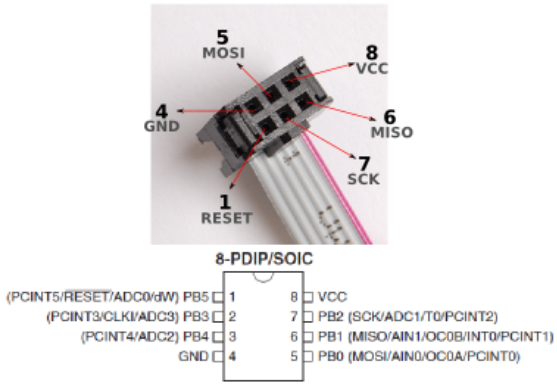
Now that we have secured the communication between our computer and the programmer lets make sure that we can communicate with our microcontroller.

First off we need to connect the MISO, MOSI, GND, VCC, RESET, and SCK pins of your microcontroller to the appropriate pins of the programmer board.

//If you are not using the attiny85 google the data sheet for your chip and connect the programmer to //the MISO, MOSI, GND, VCC, RESET, and SCK pins of your microcontroller.

In my case I put my microcontroller onto a breadboard and connected it to the programmer with a set of jumper cables





Now that all the pins are connects correctly lets see if Avrdude recognizes the microcontroller.

In Cmd type:
Avrdude -c usbtiny -p attiny85

/*
If you are not using the attiny85 enter
Avrdude -c usbtiny
A list of other compatible microcontrollers will pop-up.
In place of **attiny85** use the name of your chip
*/

Hit Enter.

```
C:\>avrdude -c usbtiny -p attiny85
avrdude: AVR device initialized and ready to accept instructions
Reading | ################################################## | 100% 0.01s
avrdude: Device signature = 0x1e930b
avrdude: safemode: Fuses OK
avrdude done.  Thank you.
```

You should receive this message
Avrdude has reported back saying that the connection has been established and the chip is ready to be programmed.

Possible Errors

Make sure that you have all the pin are connected correctly.

Atmel Studio

At this point all the hardware should be working properly. We just need to make sure that Atmel Studio can interface with Avrdude and your programmer.

This youtube video has a great guide on getting Atmel studio working. If you successfully follow the video feel free to skip to the end of this tutorial.

“Using a USBTiny Programmer with Atmel Studio”
https://www.youtube.com/watch?v=Af5P79IzcyE

First open a new “GCC C Executable Project” in Atmel Studio.

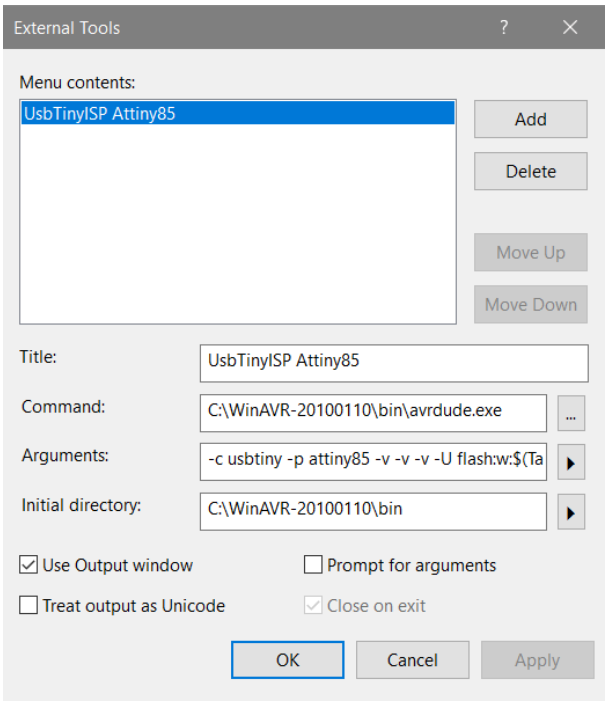
A pop-up should prompt you to select the microcontroller that you will be uploading to.

Select “attiny85”. //Or whatever model you are using.

Now before we write any code lets interface Atmel Studio with our programmer.

Go to “Tools” and hit “External tools...”

Decide on a title for you tool
For “Command” enter the PATH to you avrdude.exe file.
For “Arguments” enter: **-c usbtiny -p attiny85 -v -v -v -U flash:w:$(TargetDir)$(TargetName).hex:i**
This will instruct Atmel Studio how to upload a hex file using Avrdude.
For the “Initial directory” enter the PATH to the folder which contains the avrdude.exe file
Make sure to hit “Use Output window” and click “OK”.
The window should look like this:



At this point you should return to you main.c file. //This is the file that was automatically created and //opened when you created the new project.

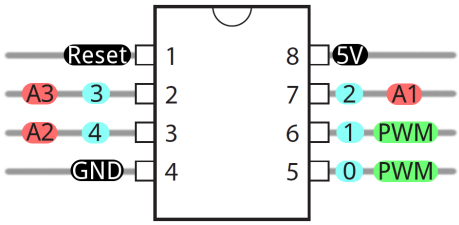
Input the following code.
//This will assigns Pin 0 on the Attiny85 as the output pin and blinks it every 100ms.
//The code may be slightly different if you are using a different microcontroller. (Try looking for example code for to blink an LED online)

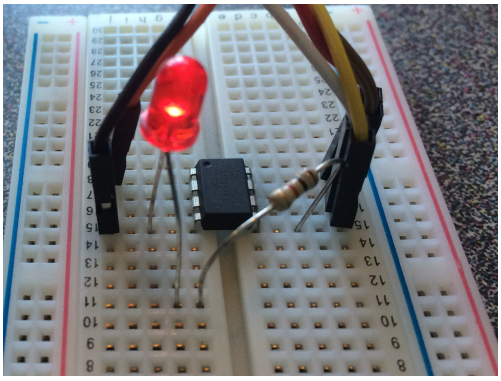
```
#include <avr/io.h>
#include <util/delay.h>

#define LED1 0

int main(void)
{
    DDRB|= (1<<LED1);
    while (1)
    {
        PORTB |= (1<<LED1);
        _delay_ms(100);
        PORTB &= ~(1<<LED1);
        _delay_ms(100);
    }
}
```

Before we upload this code make sure that you have an LED attached to pin 0.





(The light should not be on yet)

Now we need to upload the code.

Save your project.

Then under the “Build” tab hit “Build (YourProjectName)”.
This will create all the files that the microcontroller can read (the Hex files)
Any error or warning will be printed below.

To upload the compiled code click the “Tools” tab and select the external tool that you created (in my case “UsbTinyISP Attiny85”)

The code should be uploaded and Atmel Studio will display the following: After a long list of checked and uploading messages.

```
avrdude.exe done.  Thank you.
```

The LED should now blink!

Possible Errors

If the LED does not turn on but the code successfully uploaded, try flipping the LED or checking it against another power source to make sure it works. Also make sure that you have a resistor in series with it (Otherwise you may burn it out).

The End

Now that you know how to upload code, Its time for you to learn how to write it yourself. There are online tutorial for almost everything and this includes writing code for microcontrollers.

Any error messages that you may get along the way have probably been encountered by others and if you google them I am sure someone can help you see them through!

Best of luck on your future microcontroller-infused projects!

Starter's Challenge

Try writing a for loop that will make your LED change the frequency at which it blinks over time!