EE333 Intro to Microcontrollers

Lab#1 Microcontroller Basic Operation

Allan A. Douglas, Associate Professor

**Lab Overview**

In this lab, the student will build an ATmega328P based microcontroller system on a breadboard, connect basic peripherals to allow the microcontroller to boot and execute basic Arduino sketches. Students will download software and confirm that the breadboard version of the microcontroller works exactly like the printed circuit board version of the ATmega328P included with the Arduino UNO or Spark Fun RedBoard.

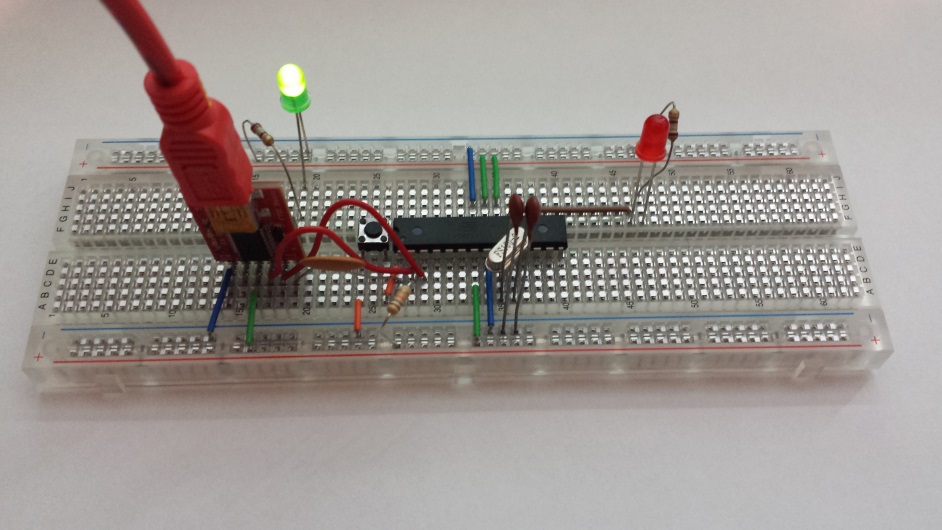
**Required Materials**

|  |  |
| --- | --- |
| Breadboard | Push button |
| ATmega328P Microcontroller | FTDL Basic Breakout board |
| ATmega328P Microcontroller Datasheet | Jumper wires |
| 16 MHz crystal oscillator | Oscilloscope |
| 2x 22pF capacitors | Arduino UNO (or Spark Fun RedBoard equivalent) |
| 10k Ohm resistors | Arduino ISE Software |
| 180 Ohm resistors | WinAVR (Extra Credit) |

**Required Work**

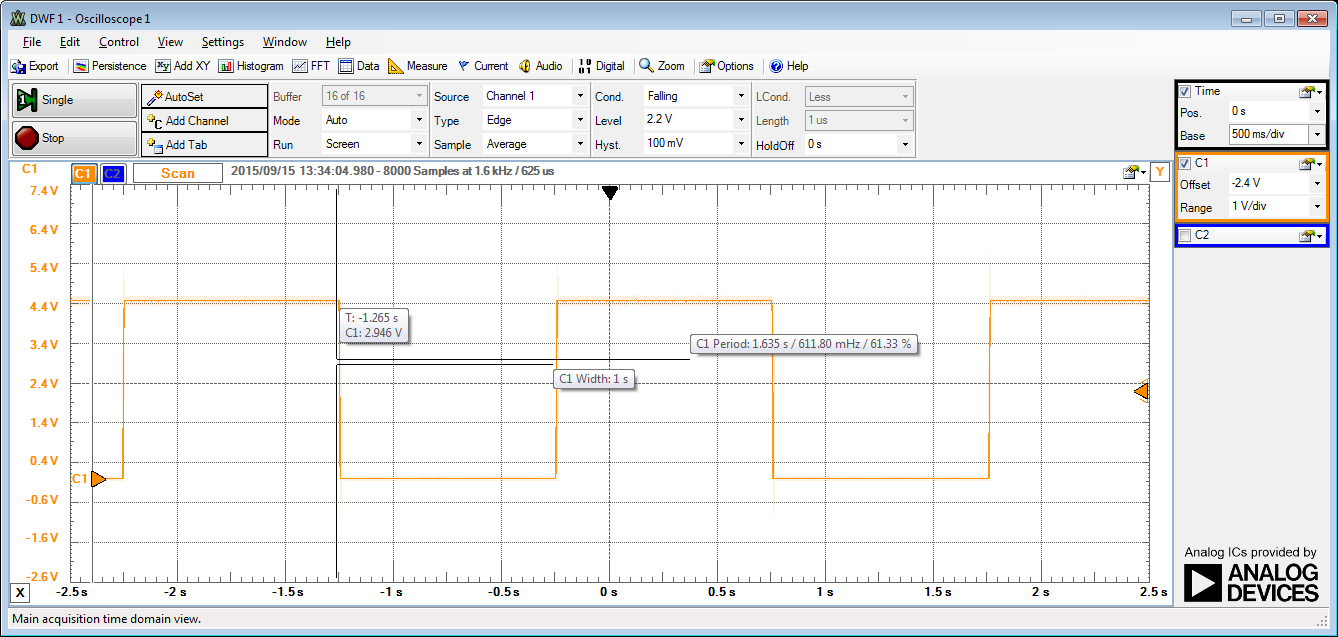
Refer to the following webpage for help building the Arduino: <http://www.jameco.com/Jameco/workshop/JamecoBuilds/arduinocircuit.html?CID=arduinobuild>

On Step 2 of the web instruction, you do not need to build the 5V power supply with the 7805T (skip parts c and d). The FTDL will provide power when it is plugged into USB. Connect the power and ground rails on your breadboard as they describe. You may also skip Step 4.

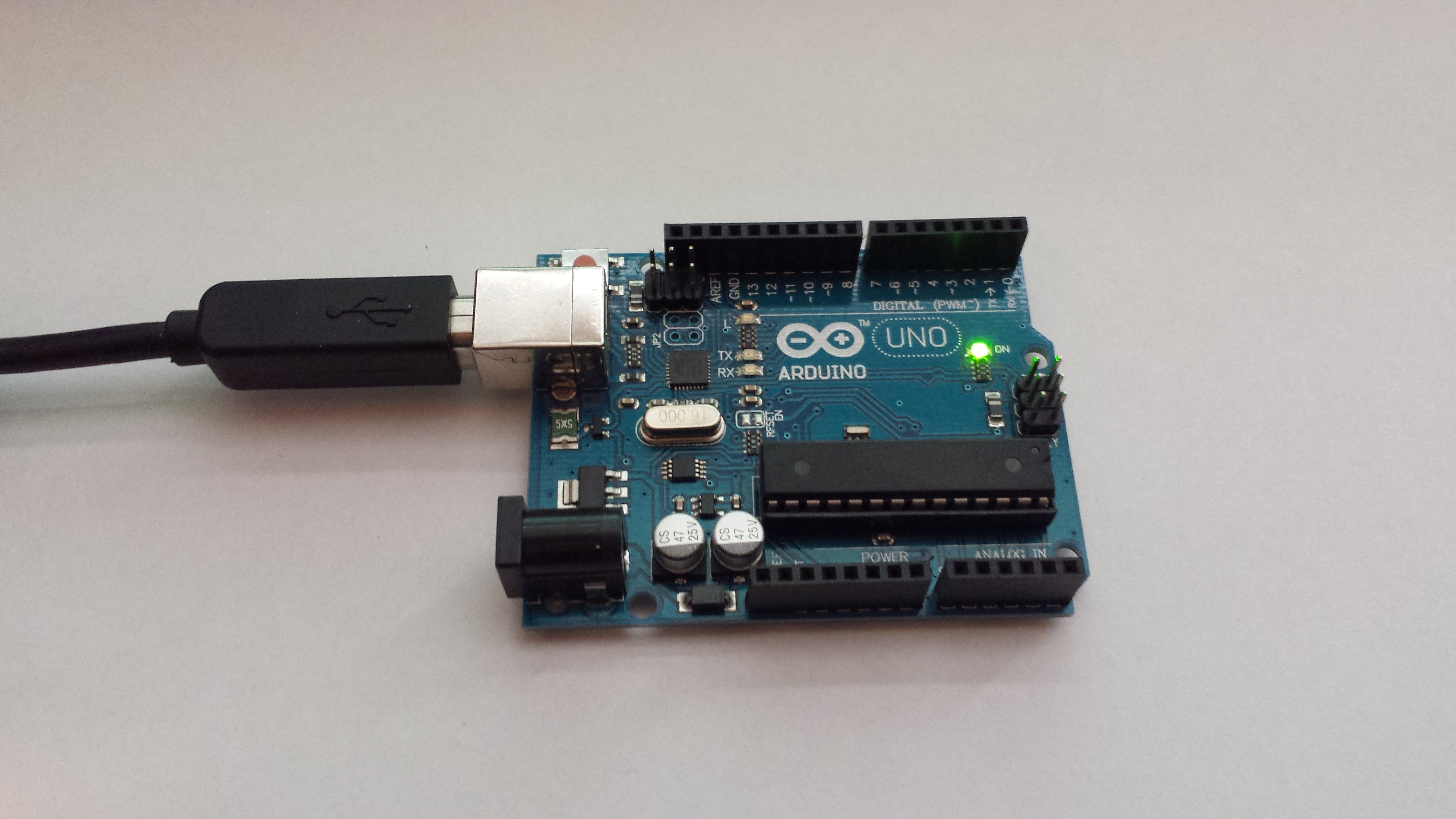


After completing this process, apply power by connecting a micro USB cable from a PC to the FTDL board. Verify the power LED is illuminated and measure the 5V power with a voltmeter.

1. Open the Arduino IDE. Select the Blink example sketch. In the Arduino IDE software Tools menu, select the Arduino board and the correct Port corresponding to your USB connection.
2. Compile and upload the sketch.
3. Verify that the LED is blinking at 0.5 Hz (on for 1 second and off for one second).



1. Verify that the Reset button causes the LED to stop blink and when you release it, the microcontroller restarts code execution.
2. Unplug the USB cable to remove power from the microcontroller. Plug in the cable again. Verify that your code begins automatically begins execution without having to download again.
3. In your Blink sketch, add a #define before the setup function and define a constant for the toggle rate. Use this constant in the delay function so that is it easy to change the blink rate.
4. Change the Blink sketch so that the LED toggle rate to 10Hz. Measure with an oscilloscope.
5. Change the Blink sketch so that the LED toggle rate to 100Hz. Measure with an oscilloscope.
6. Now, connect your fully assembled Arduino UNO (RedBoard) board to USB.



1. In the Arduino ISE, change the comm port to the UNO.
2. Program your Blink sketch into the UNO and repeat the steps 3, 7 and 8 to vary the blink rate from 0.5Hz, 10Hz, and 100Hz.
3. Research the pinMode command. Describe the options and how it can be used.
4. Research the digitalWrite command. Describe the options and how it can be used.
5. Research the delay command. Describe the options and how it can be used.

**Required Work Checklist**

□ For the breadboard microcontroller:

Picture of the working microcontroller executing the Blink sketch

Oscilloscope waveforms showing pin 9 toggling on and off at both 0.5Hz, 10Hz and 100Hz

□ For the Arduino UNO (RedBoard):

Picture of the working microcontroller executing the Blink sketch

□ Oscilloscope waveforms showing pin 9 toggling on and off at both 0.5Hz, 10Hz and 100Hz

□ Copy of your modified Blink sketch

□ pinMode command description

□ digitalWrite command description

□ delay command description

□ Write a summary of your experience with the lab. Make sure your summary answers the following questions.

What did you learn in the lab?

What was challenging about the lab?

Explanation of how the Arduino maintains your code through a power cycle. Where is your code stored?

**Extra Credit Work**

1. Implement the Blink function using C code following the example on the following webpage: <https://balau82.wordpress.com/2011/03/29/programming-arduino-uno-in-pure-c/>. Copy the C code to a text editor and save it as led.c.
2. Create a batch file in a text editor called build.bat.
3. Add commands to the batch file to compile, link, and download the code using avr-gcc, avr-objcopy, and avrdude as shown in the example. Make sure you select the correct comm port (ex. -P com5) for the avrdude command.
4. Open and command console on your PC and execute your build batch file. This will download your code. Verify operation in both the breadboard and the Arduino UNO.
5. In your own words, explain the use of bitwise operators, macros, the included header files.
6. In your own words, explain the function of avr-gcc, avr-objcopy, and avrdude executables and the options that are used.

**Extra Credit Checklist**

□ Copy of C code

□ Copy of you avr-gcc, avr-objcopy, and avrdude batch file

□ Explanation of bitwise operators, macros, and the included header files

□ Explanation of avr-gcc, avr-objcopy, and avrdude executables and the options that are used