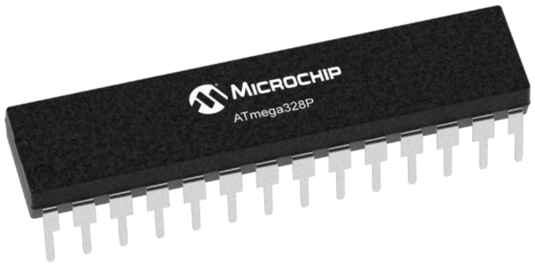
Assignment 9: Hello World

Introduction:

In the assignment, you will construct a barebones microcontroller circuit. The microcontroller you will use is the ATMega328P, which is the same microcontroller used in some of the Arduino platforms and also commonly used in a wide variety of commercial applications.



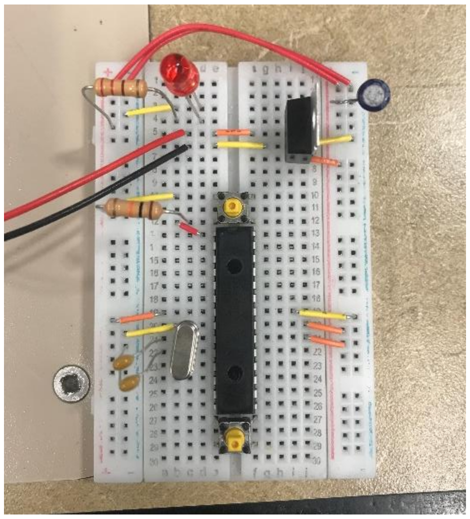
ATMega328P Microcontroller

Part 1: Building the Microcontroller Circuit on a Breadboard and Pocket Programmer Driver Installation

First, briefly browse through the datasheet for the microcontroller and familiarize yourself with its specifications and capabilities:

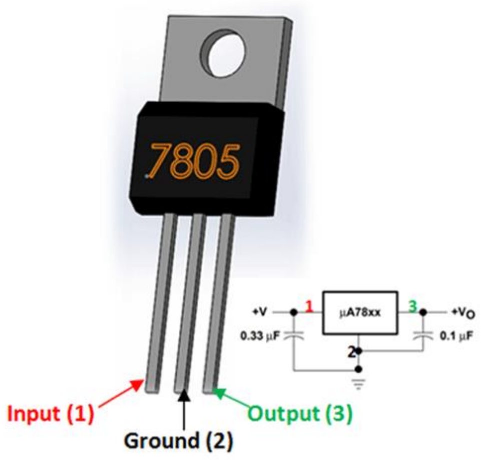
https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

One of the most well-known use cases for the ATMega328P is serving as the primary processor in the Arduino Uno. On Arduino's website, they provide a guide that shows how to construct an Arduino-compatible ATMega328P circuit on a breadboard. Your first steps should be to construct this circuit:



This circuit is powered via a 7805 voltage regulator. The role of this regulator is to ensure that the output voltage it supplies is constant regardless of the variation of the input. The data sheet for this regulator can be found here:

https://www.sparkfun.com/datasheets/Components/LM7805.pdf



https://components101.com/ics/7805-voltage-regulator-ic-pinout-datasheet

The 7805 produces a constant output voltage of 5V. However, as noted on page 3 of the datasheet, the minimum input voltage needed to produce a constant 5V output voltage is 7V. The maximum input voltage this regulator can withstand is 25V. Any input voltage within that range (7V < V < 25V) will produce a steady 5V output. You can produce suitable input voltages by using a bench supply, a 9V battery or a 12V wall supply. Be careful, when connecting these voltages to your breadboard, make sure that the microcontroller is powered via the regulator output and not directly from one of these high voltage inputs, else you will damage the chip.

Options for providing the Voltage Regulator input

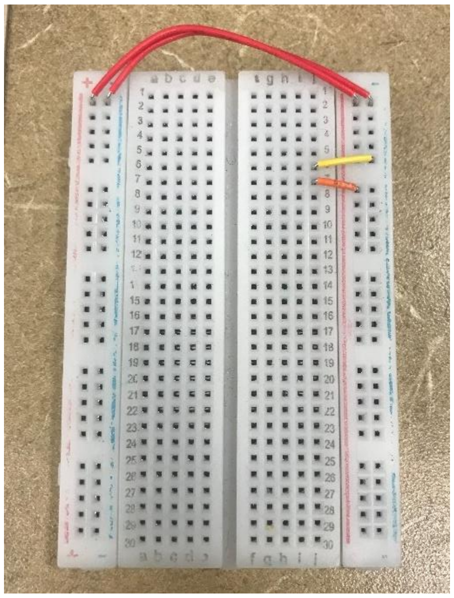
Since we are using a barebones microcontroller rather than a development platform, we need a means to load software onto the device. In order to do this, we will use an AVR programmer. This web address explains the pocket programmer that we will use for this exercise. Please follow the instructions on this website very carefully:

https://learn.sparkfun.com/tutorials/pocket-avr-programmer-hookup-guide?_ga=2.181586916.2103969426.1540442848-377410335.1540193575

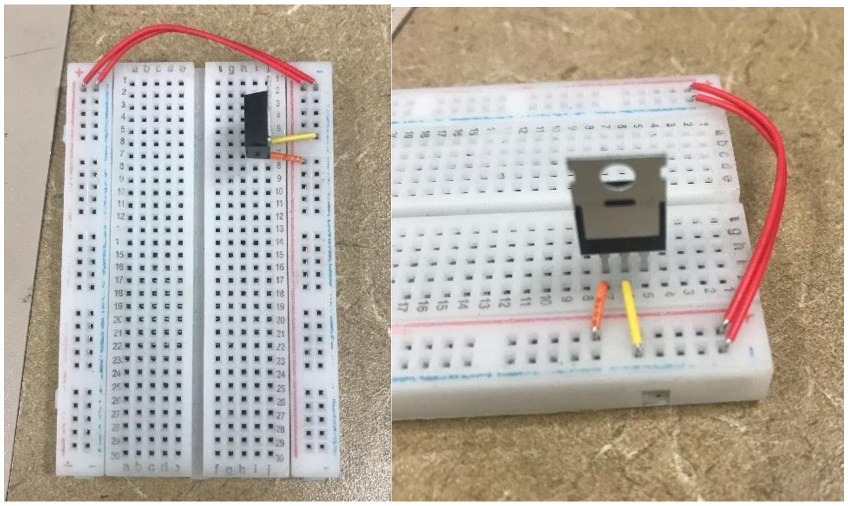
Please note that the driver installation step is critically important, and your device will not be recognized if those steps are not followed.

Now that you have the driver installed and are familiar with the voltage regulator you will be using, you can start to build the circuit. The first thing you will be building is the power supply that will supply your microcontroller 5V from a 9V battery. Note that you can use a 5V power supply in the lab to power the chip, but in order to gain full credit on this assignment, your circuit pictures will need to include this voltage regulator/9V battery setup.

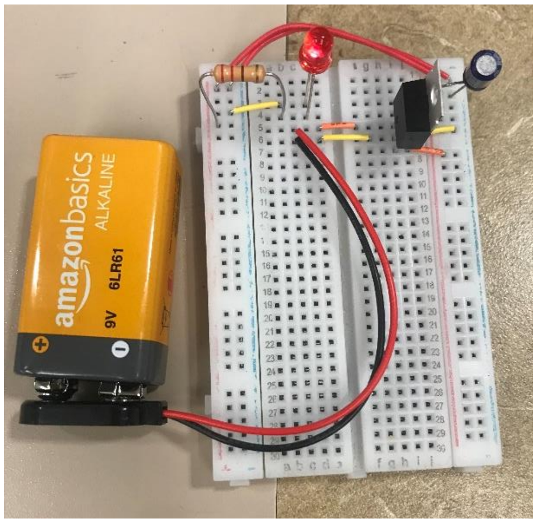
First, add two wires that connect voltage and ground to both power rails on your breadboard. Then, add power and ground wires for where your voltage regulator will be. Here is an image (yellow is ground, orange is power):



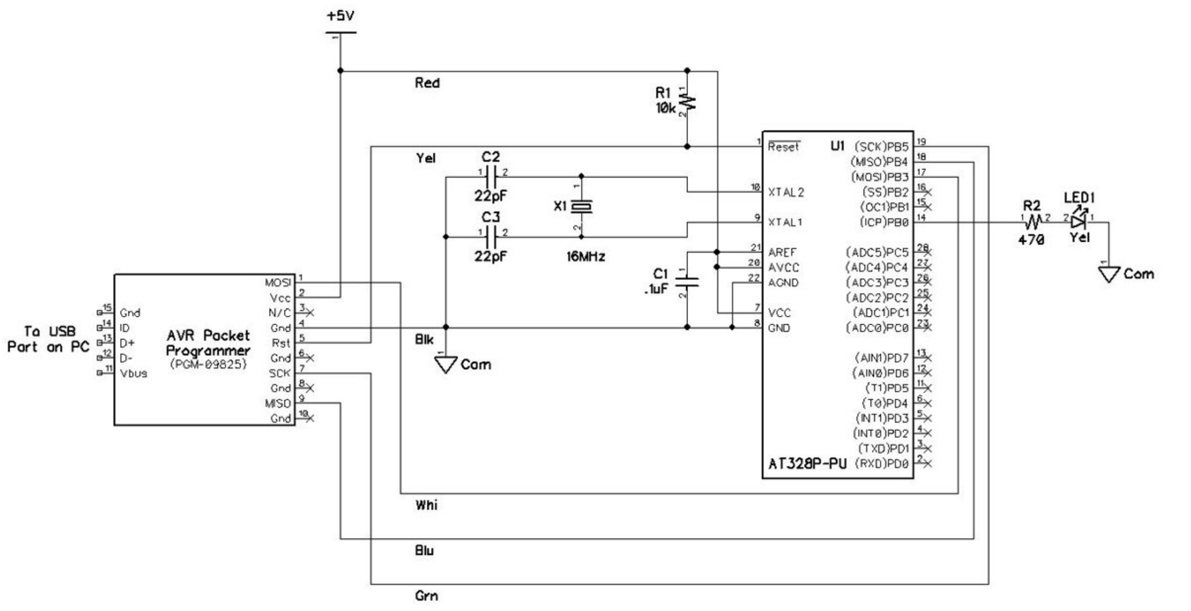
Then, you will add the voltage regulator. Place it so that if the back is to the right, the ground is in the middle prong, and the voltage is at the bottom prong. Here are some images:



Add wires to connect the unregulated power to the front of the voltage regulator. Looking from the top of the breadboard, this means the power wire will be supplied at the top prong, and ground at the middle prong. Also, add a 10 uF capacitor on the right rail in between power and ground. Another optional addition to the circuit is an LED that is powered by the regulated power supply. This is helpful because it is an easy way to see if the circuit is being powered. Here is an image:



Now that the 5V regulated voltage supply is setup, the microcontroller circuit can be setup. Here is a schematic for the circuit that you can reference throughout the rest of the tutorial.



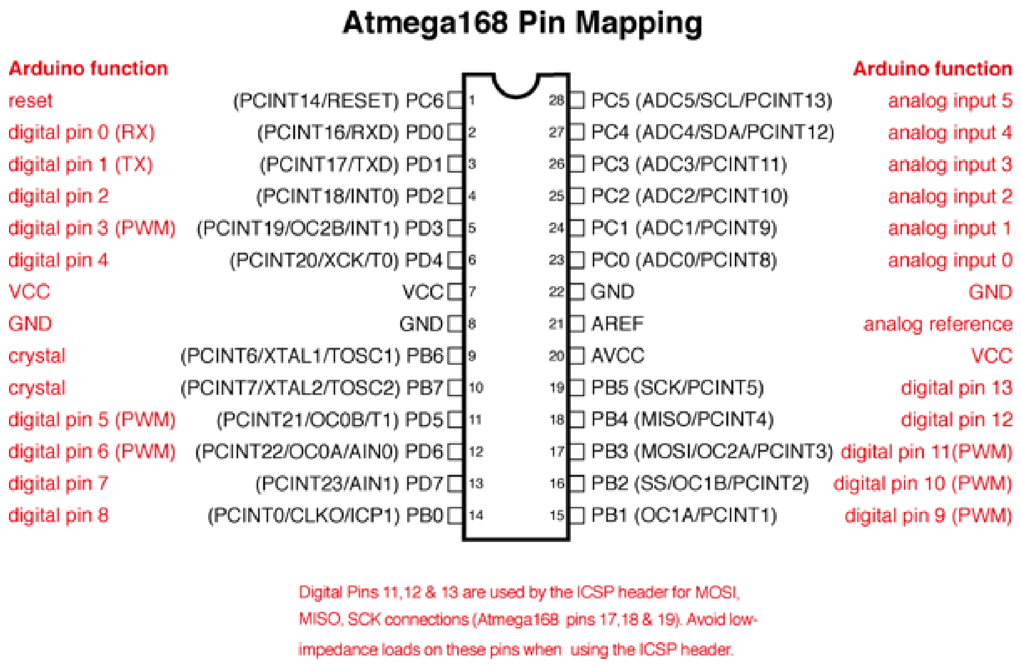
AT328P Programming Circuit with Crystal Oscillator & AVR Pocket Programmer

Image source: https://www.allaboutcircuits.com/projects/atmega328p-fuse-bits-and-an-external-crystal-oscillator/

This tutorial is based on the tutorial linked below, so feel free to use this as well:

<u>https://www.arduino.cc/en/Main/Standalone</u>

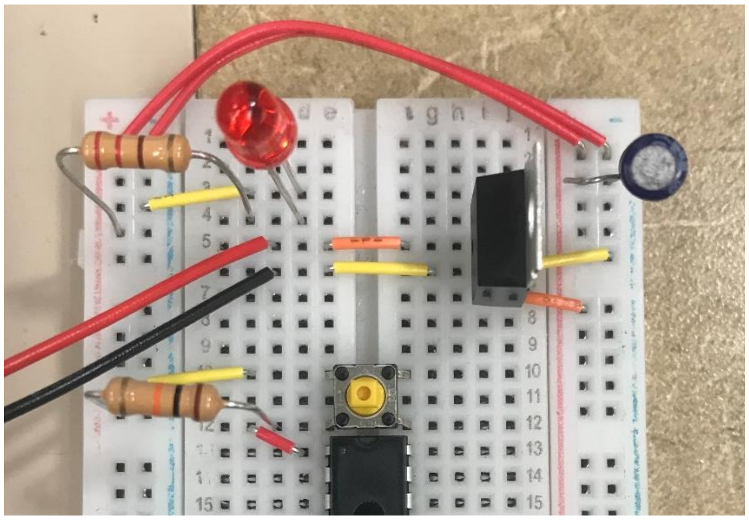
In addition, the pinout of the ATMega328P will be very helpful to reference throughout the tutorial:


Atmega168 Pin Mapping
Arduino function
reset (PCINT14/RESET) PC6 1
digital pin 0 (RX) (PCINT16/RXD) PD0 2
digital pin 1 (TX) (PCINT17/TXD) PD1 3
digital pin 2 (PCINT18/INT0) PD2 4
digital pin 3 (PWM) (PCINT19/OC2B/INT1) PD3 5
digital pin 4 (PCINT20/XCK/T0) PD4 6
VCC VCC 7
GND GND 8
crystal (PCINT6/XTAL1/TOSC1) PB6 9
crystal (PCINT7/XTAL2/TOSC2) PB7 10
digital pin 5 (PWM) (PCINT21/OC0B/T1) PD5 11
digital pin 6 (PWM) (PCINT22/OC0A/AIN0) PD6 12
digital pin 7 (PCINT23/AIN1) PD7 13
digital pin 8 (PCINT0/CLKO/ICP1) PB0 14
Arduino function
28 PC5 (ADC5/SCL/PCINT13) analog input 5
27 PC4 (ADC4/SDA/PCINT12) analog input 4
26 PC3 (ADC3/PCINT11) analog input 3
25 PC2 (ADC2/PCINT10) analog input 2
24 PC1 (ADC1/PCINT9) analog input 1
23 PC0 (ADC0/PCINT8) analog input 0
22 GND GND
21 AREF analog reference
20 AVCC VCC
19 PB5 (SCK/PCINT5) digital pin 13
18 PB4 (MISO/PCINT4) digital pin 12
17 PB3 (MOSI/OC2A/PCINT3) digital pin 11(PWM)
16 PB2 (SS/OC1B/PCINT2) digital pin 10 (PWM)
15 PB1 (OC1A/PCINT1) digital pin 9 (PWM)

Digital Pins 11,12 & 13 are used by the ICSP header for MOSI, MISO, SCK connections (Atmega168 pins 17,18 & 19). Avoid low-impedance loads on these pins when using the ICSP header.

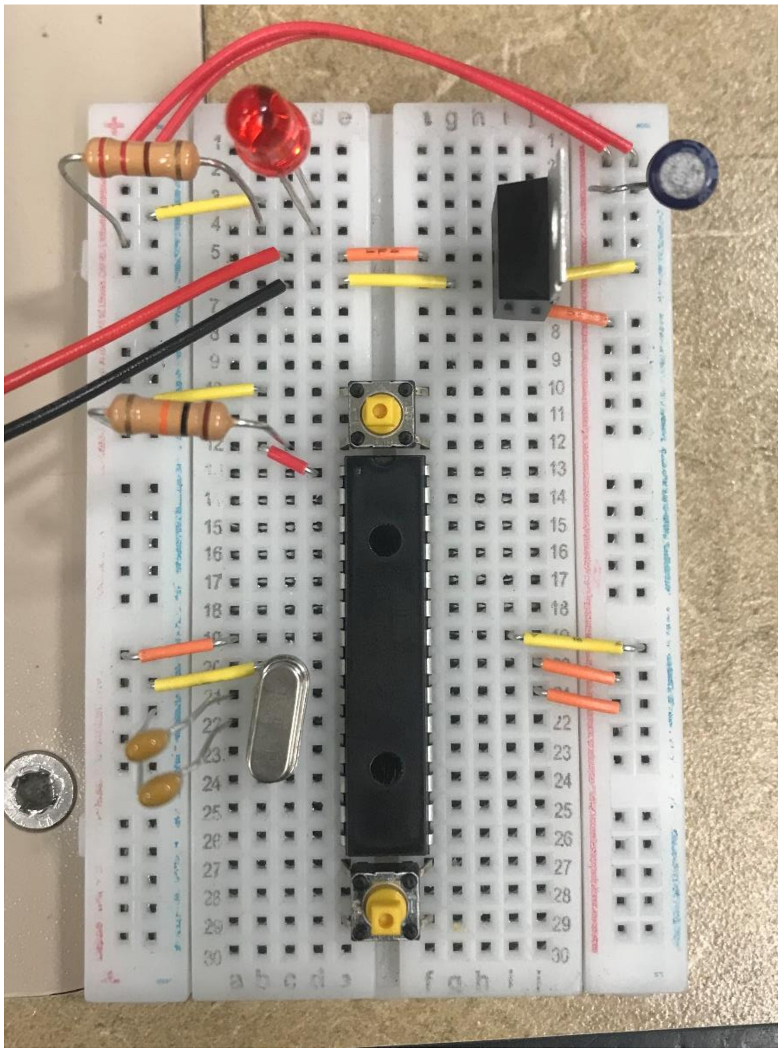
Image Source: <u>https://www.arduino.cc/en/Hacking/PinMapping168</u>

To start, connect a 10kΩ pullup resistor from the RESET pin (pin 1 on the chip) to the 5V rail. This prevents the chip from resetting itself during normal operation. The chip actually resets when pulled down to ground. Add a small tactile switch right above the chip and connect it to ground and the reset pin. This button will reset the chip when pressed. Here is an image:



Then, using the pinout for reference, connect pin 7 (VCC) to the 5V rail, pin 8 (GND) to the ground rail, pins 20 (AVCC) and 21 (AREF) to the 5V rail, and pin 22 (GND) to the ground rail.

Now we are going to add the clock source to the circuit. Place the 16 MHz external clock between pin 9 and 10, and add two 22 pF capacitors from ground to each pin of the clock. Here is an image of the completed circuit:



Part 2: Using the Pocket Programmer to Upload Software to the Chip

For this part of the assignment, you are going to learn how to upload code to the chip. In the Arduino IDE, write a program for the ATMega328P that causes the microcontroller to periodically blink an LED whenever a button is pressed. When the button is not pressed, the LED should remain off. The duration of the blinking is up to you, but it should be slow enough that you can clearly see it. The ATMega pin mapping diagram above will be very helpful in this part of the assignment.

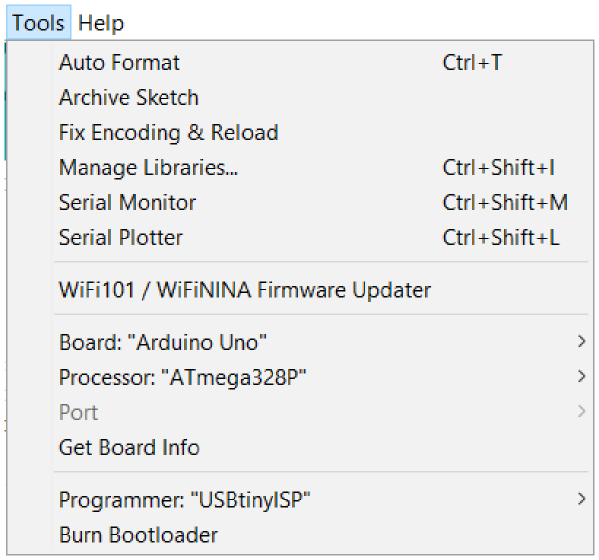
Here is a link to a tutorial that shows how to periodically alternate the voltage on an output pin between high and low voltages if you need help:

https://www.arduino.cc/reference/en/language/functions/digital-io/digitalwrite/

Once you are done with writing the software, you can compile it and upload it to your chip. VERY IMPORTANT NOTE: WHEN THE POCKET PROGRAMMER IS PLUGGED INTO YOUR COMPUTER AND HOOKED UP TO YOUR CIRCUIT, MAKE SURE THE **BATTERY IS DISCONNECTED**. If the pocket programmer is supplying power to your circuit from your computer while your battery is connected, your computer will be damaged and may not be able to be repaired.

Here are some steps to follow to upload the code to the chip once you are done with the software. You can also find these steps in the same tutorial that has the drivers for the pocket programmer.

1. Using the schematic shown previously called “AT328P Programming Circuit with Crystal Oscillator & AVR Pocket Programmer” to connect the pocket programmer to the ATMega. Note: the pins on the pocket programmer are labeled on the bottom of the device.
2. In **Tools -> Board**, make sure you select “Arduino UNO”, which has contains the ATMega328P as its microcontroller.
3. In **Tools -> Programmer**, make sure you select “USBtinyISP”.


Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Processor: "ATmega328P"
Port
Get Board Info
Programmer: "USBtinyISP"
Burn Bootloader

*Note: It is okay if you do not have the processor option on this menu

4. Once these configurations have been made, you can click the teal checkmark at the top of left corner of the IDE to compile the project.
5. If it compiles, go to **Tools -> Burn Bootloader.**
6. Upload the code to the chip by clicking **Sketch -> Upload Using Programmer.**

Once you get the simply blinking function to work, amplify the code so that when you push the button, it makes three LEDs continuously take turns flashing. Look at the demo videos to see what the final submission should look like:

One light blink:

https://drive.google.com/file/d/1nAAcLuTBlWDzb7xvu-nf4Q3hN33Le3da/view?usp=sharing

Three light blink:

https://drive.google.com/file/d/1p0WA9SL3t09XhF3sPA3AWg-S-OPyzdpq/view?usp=sharing

Submission:

- For this assignment, you must submit your code for both the one light and three light blink.
- Also upload a video demo for both the one light and three light blink that demonstrates its full functionality.