Skill Demo 3: Microcontrollers

v1.6

Resources

Arduino Mega Reference Sheet
Atmel ATmega2560 full datasheet
Arduino Getting Started Guide

Overview

A microcontroller is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Program memory in the form of flash, EEPROM, or one-time programmable ROM is also often included on chip, as well as a typically small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications.

This semester we will be using an Arduino-compatible microcontroller development board, the Arduino Mega. The microcontroller onboard is an Atmel ATmega328p, an 8-bit processor running at 16 MHz, with 2KB of RAM, and 32KB of Flash memory.

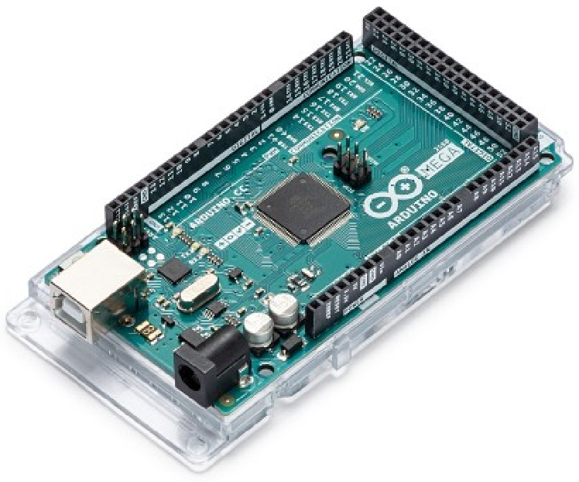


Figure 1: Arduino Mega

While we could write all of our code from scratch for the ATmega2560, it's easier to get started using the Arduino IDE, which includes all of the tools and many common libraries. It also knows how to program the Arduino boards over a USB cable. The 'Arduino code' used by is actually C/C++ code within a simplified execution structure.

So what can I do with this thing?

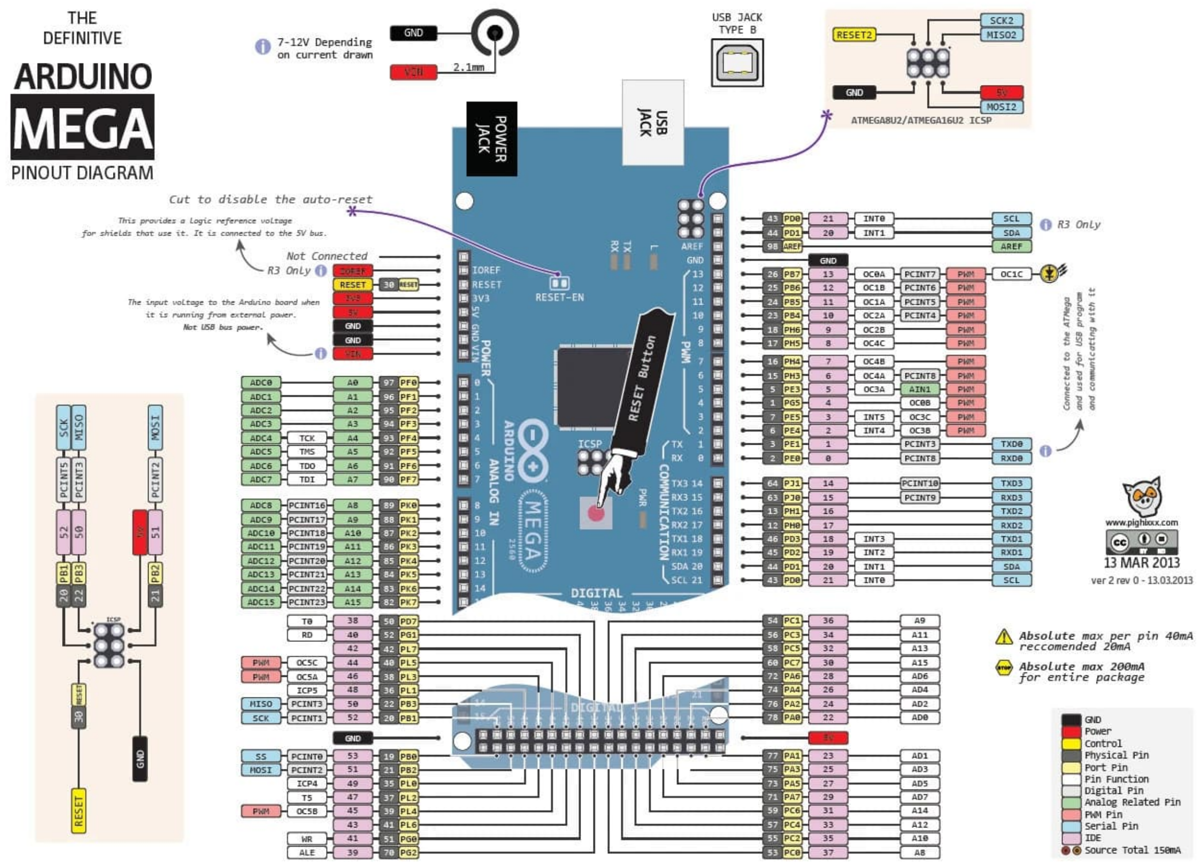

THE DEFINITIVE ARDUINO MEGA PINOUT DIAGRAM
7-12V Depending on current drawn
USB JACK TYPE B
ATMEGA8U2/ATMEGA16U2 ICSP
Cut to disable the auto-reset
Absolute max per pin 40mA reccomended 20mA
Absolute max 200mA for entire package
www.pighixxx.com
13 MAR 2013
ver 2 rev 0 - 13.03.2013

Figure 2: ATmega2560 Peripherals

Microcontrollers include the processor and memory that all computers share. What makes them useful is that their hardware is designed for directly interfacing with sensors and other devices. The ATMega2560 has built-in peripherals that are able to read digital and analog signals, and output digital signals, including RS232, I2C, and SPI. Its main strengths are its connectivity, low power consumption (<10mA, less than your LEDs), low power modes, and low price (<$4).

Figure 2 shows what can be done with each pin. The color code links pin function to category on the right. Note that each pin may perform multiple functions depending on the configuration of the processor. For example Pin A8 in the middle left can be used as a digital input or output, analog input, an interrupt pin. Pin 21 can be used as a digital input or output, or the SCL line of an I2C bus (a communication standard for multiple devices on the same wires).

1: Learn not to damage your Mega

Use the datasheet for the ATmega2560 microcontroller that we're using (or other resources) to answer the following:

1.1: There are pins specifically for powering the ATmega2560. In their documentation, Atmel refers to this power supply as "VCC", relative to the "GND" pin. Based on *Section 32: Electrical Characteristics* in the datasheet, what is the voltage range for the processor to operate normally?

1.2: The microcontroller is soldered down the Arduino Mega circuit board, and has electrical connections to the Mega's 0.1" headers. Using the schematic for the Arduino Mega, what pin(s) on the 18x2 pin Mega header (labeled XIO) is connected to the 4 positive power input pins (VCC) of the ATmega2560? (You might have to follow a power rail.)

1.3 If you power the Mega off a battery pack plugged into the barrel connector (X1), there is a component that converts the battery voltage to 5V. This device has an input, 2 pins connected to the output, and a ground. Find this component on the schematic and list its part number.

1.4 Most of the ATmega2560 pins can be used as general-purpose input pins or output pins (GPIO). What's the maximum and minimum safe voltage for these pins?

1.5: What is likely to happen if we exceed one of these input voltage ratings?

1.6: The GPIO pins can be set in input mode, reading voltages as "high" or " low". As you'd expect, 0V will read as a '0', and +5V will read as '1'. What value will you read if you supply an input pin with each of the following voltages? [0.8V, 1.2V, 3.3V, 4.5V]
(Hint: look in the "Electrical Characteristics section of the datasheet.)

1.7: When in input mode, the pins can have a "pull-up" or "pull-down" resistor activated to keep the voltage near VDD or GND when not otherwise driven. Program your Arduino Mega to enable the pullup resistor, and short that pin to ground with a multimeter in current measurement mode. How much current is passing through? What must the resistance of that internal pull-up resistor be? What does the datasheet say it should be?
(Hint: You can enable a pullup using the Arduino libraries using digitalWrite() and/or pinMode(), as described in the Arduino Digital Pin section.)

1.8: The GPIO pins can also be set to output "high" and "low", as you've done in previous skill demos. When GPIO is set to output 'high', what's the minimum voltage guaranteed on the pin? How much current can we draw from this pin? Is it enough for our LEDs?

Skill Demo 3: Microcontrollers

NAME____________________ GTID____________________
☐ CS3651A
☐ CS3651B

2: Write a simple program

Next, use a tactile switch (or other switch) to provide input to a digital I/O of your Arduino. You can do this using digitalRead() and a pin with an internal or external pullup or pulldown resistor.

Write a program to *continuously* read in the state of the switch, and light up an LED to replay the switch's state with a three second delay. In other words, when you press the switch in a pattern, the LED should blink the same pattern three seconds later. This 'delayed playback' needs to be running continuously, so that you can start tapping out a pattern *at any time including while playing back earlier presses*.

You may want to look at other Arduino sample sketches or the Arduino tutorials for examples of how to use digitalRead to read the state of a tactile switch, or the Arduino reference for the API for the Arduino core library. At a minimum, your program needs to mirror the state of the switch with a three second delay, and a time resolution of 100 ms, but you should be able to implement much higher resolution easily. You can choose how to store the state of the switch internally. We promise that it's possible to implement this without threads.

As part of your program, use the Serial.print() functionality, and use it to print some debugging information from your program. At a minimum, print when the tactile switch is pressed or released. Show this information on your computer as part of the sign-off.

Arduino Program structure

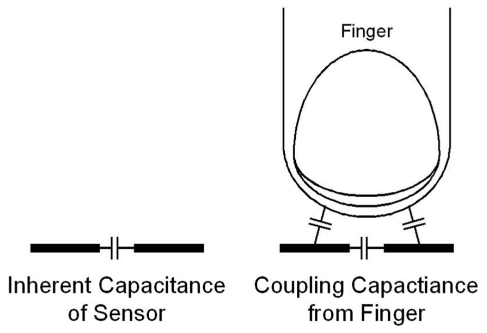
An Arduino program consists of a Setup() function and a Loop() function. The Setup() function contains all the code that is executed once at the beginning of the program, like initial pinMode() calls. This usually includes setting up the various parts of the microcontroller and initializing variables. The Loop() function is called repeatedly as fast as possible. This is where the main part of your executing code should be located.

Hints:

- Use the Arduino built-in serial terminal to monitor the output from the Mega.
- The delay() function will wait a specified number of milliseconds.
- You can declare global variables, so long as you don't fill up memory with them.

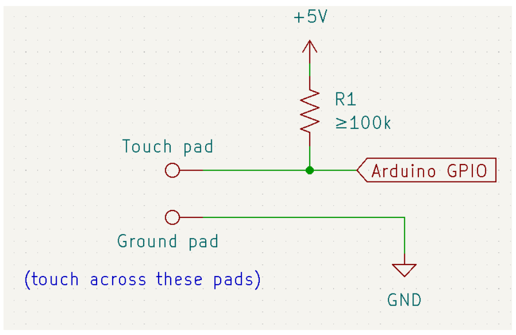
Sign-off initials:________________ Date:____________

3: Detect a touch


Finger
Inherent Capacitance of Sensor
Coupling Capactiance from Finger

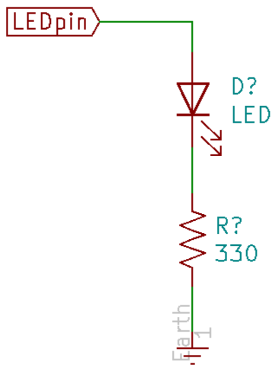
Capactitive sensing is using the change in capacitance on a conductor to detect changes in the environment of the conductor. The capacitance can be determined by measuring how long it takes to charge a conductor after completely discharging it. One common application of capacitive sensing is to create touch buttons without requiring any external components.

GPIO capacitive sensing for touch



This program is a little more involved. We are going to measure how long it takes to charge the wire connected to a pin (and whatever is connected) through a high resistance. This will let us build a touchpad that acts like a switch when a person touches the wire, or even gets very close to it.

The circuit consists of a high value pull-up resistor connected to a touch pad (some wire) and a GPIO pin. The touch pad is set up such that a finger touch touches both the pad and ground. The algorithm is:



Setup:

1) Set LED pin to output.
2) Set the touch pin to output.

Loop:

1) Output low on the touch pin to discharge the touchpad.
2) Record time t1.
3) Set the touch pin to input.
4) Wait until you read HIGH on the touch input.
5) Record time t2.
6) If t2-t1 is greater than a predetermined threshold, light the LED.
7) Do this about 100 times and average the values to get stable output.

Notes:

Use micros() for reading t1 and t2.
Use a 100K resistor or larger for R1.
The larger R1 is, the slow the charge times will be.

Build a program that senses touch on a wire while the user is also touching a ground wire. Light up an LED when the user touches the touch pin.

Sign-off initials:________________ Date:____________