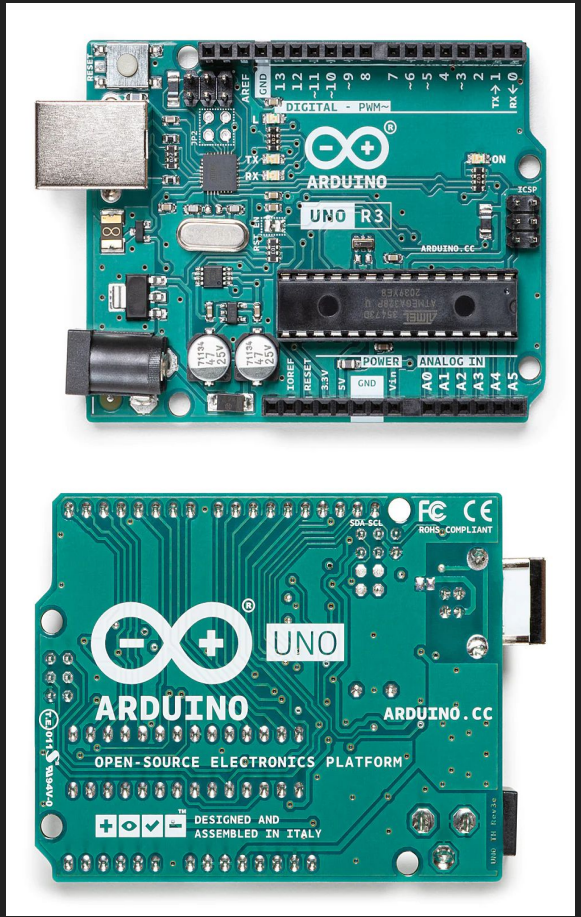
02 - Lab Introduction

CEG 4330/6330 - Microprocessor-Based Embedded Systems

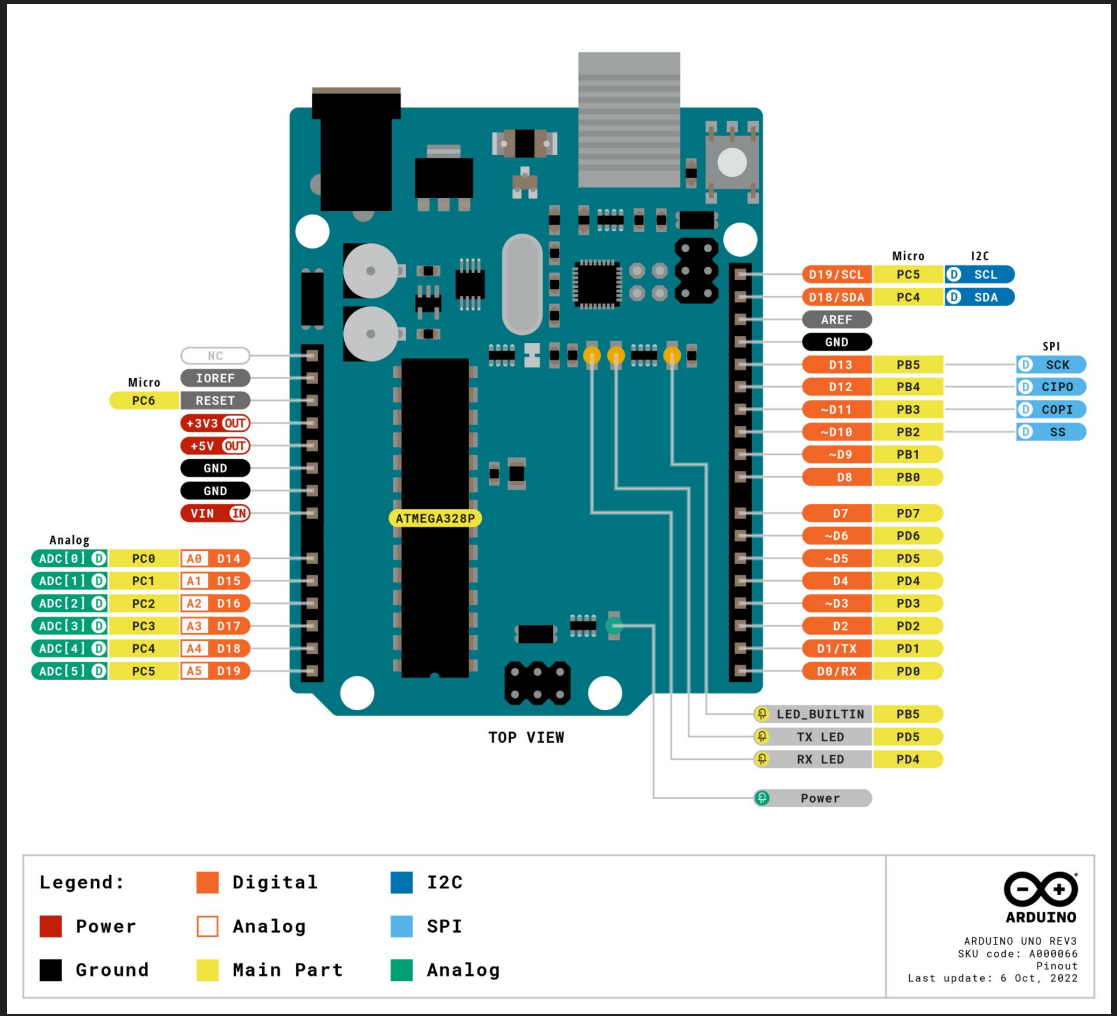
Max Gilson

Arduino Uno

- Open source
  - The design is publicly available for anyone to redesign and modify code
- Small form factor
  - 68.6 mm x 53.4 mm
  - 25 g
- Affordable and abundant
  - Easy to replace if broken
  - Use multiple for many simultaneous projects
- Based on Atmel ATmega328 processor



Pins and I/O (cont.)



Additional Specifications

- 8-bit RISC processor
- 32KB of flash memory
- Three timers with separate prescalers:
  - Timer 1: 16-bit resolution with output capture, input capture, PWM
  - Timers 0 and 2: 8-bit resolution with output capture and PWM
- 10-bit resolution analog to digital converter (ADC)
- UART, I2C, SPI communication protocols
- Data sheet:
  - https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf
- Programming reference documentation:
  - https://www.arduino.cc/reference/en/
- Arduino IDE and example code
  - https://www.arduino.cc/en/software

Default libraries

- After installing Arduino IDE, under c:\Program Files (x86)\Arduino\hardware
  - arduino\cores\arduino contains source codes for functions listed on the language reference page
  - tools\avr\avr\include\avr contains system header files (e.g., iom328p.h)

Pins and I/O

- Digital I/O Pins (14 total)
  - pinMode(pin, mode)
    - configures the specified pin to behave either as an input or output
      - pin: the Arduino pin number to set the mode of
      - mode: INPUT, OUTPUT, or INPUT_PULLUP
  - digitalWrite(pin, value)
    - set a HIGH or a LOW value to a digital output pin
    - enable (HIGH) or disable (LOW) the internal pullup on the input pin
      - pin: the Arduino pin number.
      - value: HIGH or LOW.
  - digitalRead(pin)
    - reads the value from a specified digital pin, either HIGH or LOW
      - pin: the Arduino pin number you want to read
      - returns: HIGH or LOW
  - analogWrite(pin, value)
    - writes an analog value (PWM wave) to a pin
      - pin: the Arduino pin to write to. Allowed data types: int.
      - value: the duty cycle: between 0 (always off) and 255 (always on). Allowed data types: int.
  - Communication with the PC in serial monitor uses digital pins 0 and 1

Pins and I/O (cont.)

- Analog Input Pins (6 total)
  - analogRead(pin)
    - reads the value from the specified analog pin
      - pin: the name of the analog input pin to read from
      - returns: analog reading on the pin
  - Arduino Uno: 10-bit resolution mapped to 5V operating voltage
    - 0V = 0
    - 0.25V = 51
    - 5V = 1023

Blink LED - Example Code

```
// Code from http://arduino.cc/en/Reference/DigitalWrite
int ledPin = 13; // LED connected to digital pin 13
                 // LED_BUILTIN
void setup()
{
        pinMode(ledPin, OUTPUT); // sets the digital pin as output
}

void loop()
{
        digitalWrite(ledPin, HIGH); // sets the LED on
        delay(1000); // waits for 1000 milliseconds
        digitalWrite(ledPin, LOW); // sets the LED off
        delay(1000); // waits for 1000 milliseconds
}
```

Read Analog Voltage - Example Code

```
// Code from http://arduino.cc/en/Reference/AnalogRead
int analogPin = 3; // potentiometer connected to analog pin 3
int val = 0; // variable to store the value read

void setup()
{
	Serial.begin(9600); // setup serial
}

void loop()
{
	val = analogRead(analogPin); // read the input pin
	Serial.println(val); // debug value
}
```

Output Analog Voltage - Example Code

```
// Code from http://learn.adafruit.com/adafruit-arduino-lesson-10-making-sounds/playing-a-scale */
int speakerPin = 12;
int numTones = 8;
int tones[] = {261, 294, 330, 349, 392, 440, 494, 523};
//                  C     D     E      F      G     A      B    High C
void setup()
{
        for (int i = 0; i < numTones; i++)
        {
                tone(speakerPin, tones[i]);
                delay(500);
        }

        noTone(speakerPin);
}
void loop() {  }
```

Main Function

- Main function is already provided ->
- Write your own setup() and loop()
- setup() only runs once
- loop() runs in an infinite loop

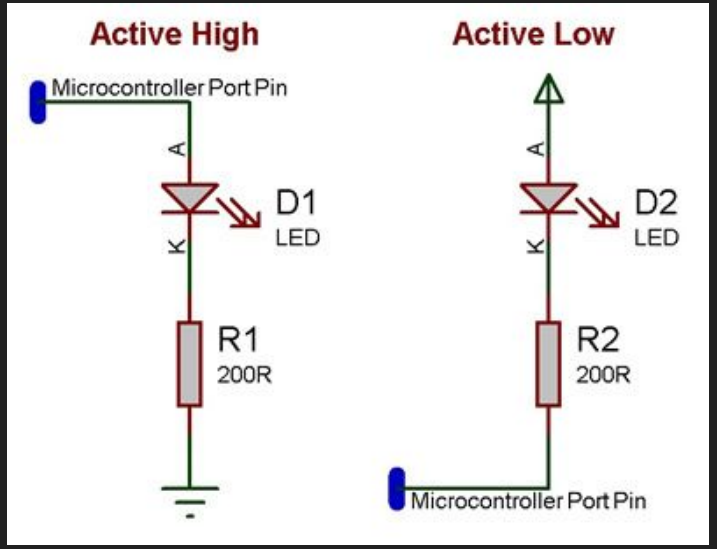
```
#include <Arduino.h>

int main(void)
{
	init();

#if defined(USBCON)
	USBDevice.attach();
#endif

	setup();

	for (;;)
	{
		loop();
		if (serialEventRun) serialEventRun();
	}

	return 0;
}
```

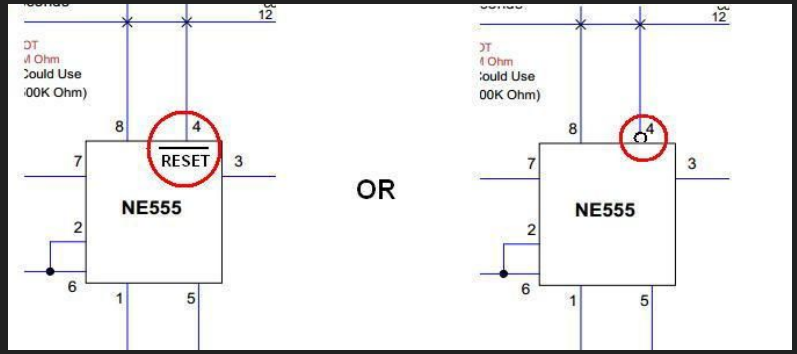
Programming with Arduino

- Read/write registers directly in high level languages
- Compile for many different platforms (avr-gcc)
  - Arduino Uno, Mega, Mini, etc.
- Custom plugins
  - Keypad, infrared sensors, sonar sensors, etc.

Active High / Active Low

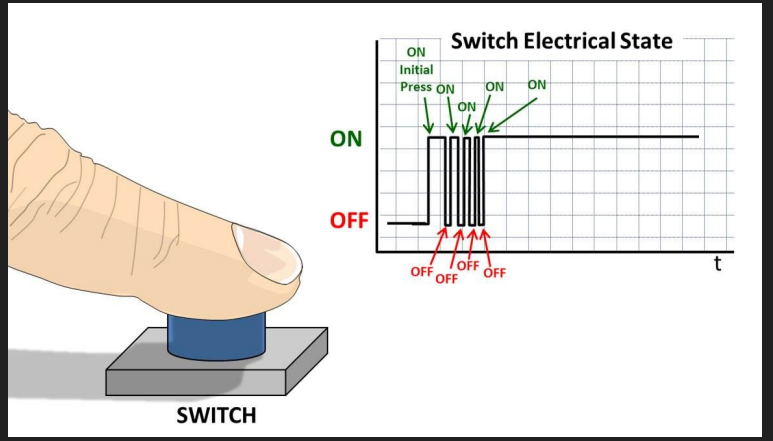
- Describes how a pin or device is activated
- Active High - you must connect the pin to HIGH (3.3V or 5V usually) to activate it
- Active Low - you must "pull" that pin LOW by connecting it to ground to activate it
- Active Low is usually described with an or a ○
- For example a chip enable pin labeled $\overline{CE}$ will require you to connect the pin to ground to enable the chip

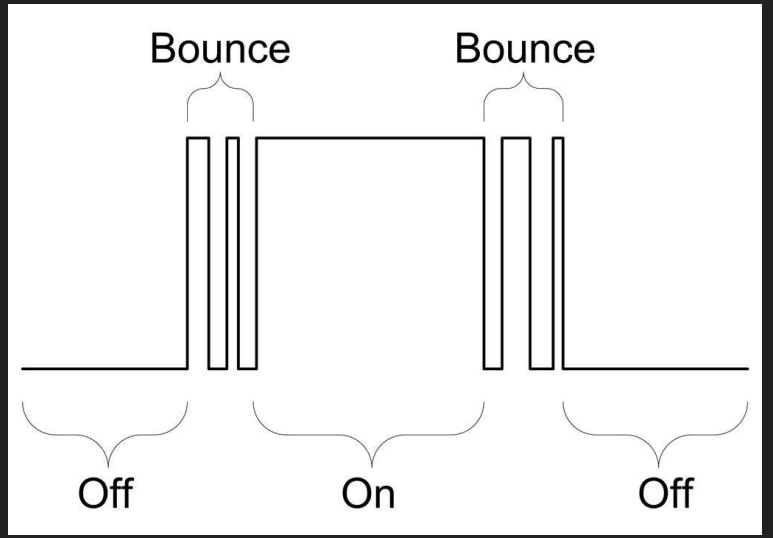




Bounce / Debounce

- When you press a switch it usually does not open/close perfectly
- The mechanical parts can “bounce” resulting in a noisy switching signal
- To fix this you must implement some debouncing
  - This can be accomplished by adding components (RC filter) or by software





Debounce - Example Code

```
/*https://www.arduino.cc/en/Tutorial/BuiltInExamples/Debounce*/

// constants won't change. They're used here to set pin numbers:
const int buttonPin = 2;  // the number of the pushbutton pin
const int ledPin = 13;    // the number of the LED pin

// Variables will change:
int ledState = HIGH;         // the current state of the output pin
int buttonState;             // the current reading from the input pin
int lastButtonState = LOW;  // the previous reading from the input pin

// the following variables are unsigned longs because the time,
        measured in
// milliseconds, will quickly become a bigger number than can be
        stored in an int.
unsigned long lastDebounceTime = 0;  // the last time the output pin
        was toggled
unsigned long debounceDelay = 50;    // the debounce time; increase
        if the output flickers

void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(ledPin, OUTPUT);

  // set initial LED state
  digitalWrite(ledPin, ledState);
}
```

```
void loop() {
  // read the state of the switch into a local variable:
  int reading = digitalRead(buttonPin);

  // check to see if you just pressed the button
  // (i.e. the input went from LOW to HIGH), and you've waited long enough
  // since the last press to ignore any noise:

  // If the switch changed, due to noise or pressing:
  if (reading != lastButtonState) {
    // reset the debouncing timer
    lastDebounceTime = millis();
  }

  if ((millis() - lastDebounceTime) > debounceDelay) {
    // whatever the reading is at, it's been there for longer than the debounce
    // delay, so take it as the actual current state:

    // if the button state has changed:
    if (reading != buttonState) {
      buttonState = reading;

      // only toggle the LED if the new button state is HIGH
      if (buttonState == HIGH) {
        ledState = !ledState;
      }
    }
  }

  // set the LED:
  digitalWrite(ledPin, ledState);

  // save the reading. Next time through the loop, it'll be the lastButtonState:
  lastButtonState = reading;
}
```

LED Circuit

- Use the example sketch “Fade” to fade an LED with the following circuit
- A resistor will be required
- To find the best resistance use the formula:

$$R = \frac{V_s - V_f}{I_f}$$

where R is the resistance, V_s is the supply voltage, V_f is the LED forward voltage, I_f is the desired (or max) LED current