Pulse-Width Modulation: Simulating variable DC output

ME 120
Mechanical and Materials Engineering
Portland State University
http://web.cecs.pdx.edu/~me120
Motivation

- Arduino Uno boards do not have arbitrary voltage output
- Pulse-Width Modulation (PWM) is a common technique for supplying variable power to “slow” electrical devices such as LEDs and DC motors
- PWM is easy to implement and greatly extends the range of control applications with microcontrollers in general and Arduinos in particular
Motivation

- Arduino Uno boards do not have arbitrary voltage output
- Pulse-Width Modulation (PWM) is a common technique for supplying variable power to “slow” electrical devices such as LEDs and DC motors
- PWM is easy to implement and greatly extends the range of control applications with microcontrollers in general and Arduinos in particular

Analog output pins?
Motivation

• Pulse-Width Modulation (PWM) is a common technique for supplying variable power to “slow” electrical devices such as LEDs and DC motors

• PWM is easy to implement and greatly extends the range of control applications with microcontrollers in general and Arduinos in particular

• “Pulse Width Modulation” $\rightarrow$ modulate the pulse width
PWM is a variable width pulse train

- The frequency of pulses is fixed
- The width of the pulse is variable

The ratio $\tau_0/\tau_c$ is called the duty cycle
PWM can act as a variable voltage

- If a PWM signal is supplied to a “slow” device, the effective voltage is

\[ V_{\text{eff}} = V_s \frac{0}{t_c} \]

- Examples of “slow” devices
  - DC motors: because of inertia and inductive energy storage
  - LED: because our eyes are slow

- “Slow” means that the frequency of the PWM pulse train is much faster than the response time of the device
Varying the duty cycle produces variable $V_{\text{eff}}$
**analogWrite(...) produces variable $V_{eff}$**

\[ V_s = 5V \]

\[ V_{eff} = 1.25 \text{ V} \]

\[ V_{eff} = 2.50 \text{ V} \]

\[ V_{eff} = 3.75 \text{ V} \]

---

http://arduino.cc/en/Reference/AnalogWrite
http://arduino.cc/en/Tutorial/PWM
Arduino Uno Pins 3, 5, 6, 9, 10, 11 for PWM

- The ~ before the pin number indicates PWM capability

http://arduino.cc/en/Reference/AnalogWrite
http://arduino.cc/en/Tutorial/PWM
PWM output is

\texttt{analogWrite( pin, dutyCycle )}

- \texttt{pin} = one of 3, 5, 6, 9, 10, 11
- \texttt{dutyCycle} is an unsigned 8-bit value
  - \( 0 \leq \text{dutyCycle} \leq 255 \)

```c
int PWM_pin = 5;  // Digital I/O pin must have PWM capability
                  // Pins 3, 5, 6, 9, 10, 11 on Arduino Uno can do PWM
void setup() {
    pinMode(PWM_pin, OUTPUT);  // Configure I/O pin for high current output
}

void loop() {
    int duty = 127;  // Duty cycle must be in range 0 <= duty <= 255
    analogWrite(PWM_pin, duty);  // Adjust duty cycle of output pin
}
```

http://arduino.cc/en/Reference/AnalogWrite
http://arduino.cc/en/Tutorial/PWM
Example: PWM control of LED brightness

• Connect a potentiometer to Analog Pin A1
• Connect an LED circuit to Digital Pin 10
• Write a code that links turning the knob of the potentiometer to the light intensity of the LED
Example: PWM control of LED brightness

- Connect a potentiometer to Analog Pin A1
- Connect an LED circuit to Digital Pin 10
- Write a code that links turning the knob of the potentiometer to the light intensity of the LED
//   File: LED_dimmer.ino
//
//   Use a potentiometer to control the brightness of an LED.
//   Voltage supplied to the LED is a PWM signal w/ variable duty cycle.

int LED_pin = 10;    //  Digital I/O pin must have PWM capability
                     //  Pins 3, 5, 6, 9, 10, 11 on Arduino Uno can do PWM
void setup() {
    pinMode(LED_pin, OUTPUT);  // Configure I/O pin for high current output
    Serial.begin(9600);        // Open serial monitor for diagnostic messages
}

void loop() {

    int duty, pot_reading, pot_pin=1;

    pot_reading = analogRead(pot_pin);    // Get potentiometer setting
    duty = map(pot_reading, 0, 1023, 0, 255);  // map 0-1023 to 0-255
    duty = constrain(duty, 0, 255);        // Make sure 0 <= duty <= 255
    analogWrite(LED_pin, duty);            // Adjust duty cycle of output pin

    // -- Print potentiometer reading and scaled delay as a diagnostic
    //    Printing values does not affect operation of the code.
    Serial.print("Potentiometer = ");
    Serial.print(pot_reading);
    Serial.print(" Duty cycle = ");
    Serial.println(duty);
}