Embedded Systems

Prof. Dr. Volker Strumpen

Rhine-Waal University of Applied Sciences

Winter Semester 2019

Syllabus

No	Date	Topic
1	10/2	Introduction
2	10/9	Facing the World
3	10/16	Sensors
4	10/23	Actuators
5	10/30	AVR Architecture
6	11/6	MCU System Architecture
7	11/13	C Programming
8	11/20	Assembly Programming
9	11/27	Timers and Interrupts
10	12/4	Communication
11	12/11	Real-Time Systems
12	12/18	Scheduling
13	1/8	Modeling
14	1/15	Model Checking

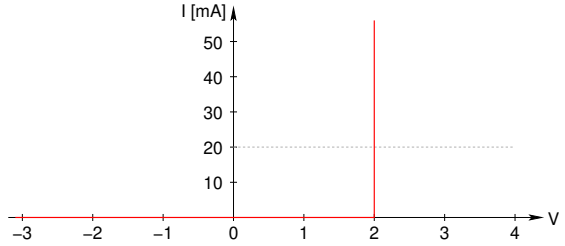
Outline

Models of Actuators

We use actuators to alter a physical quantity. As for sensors, proper use of an actuator relies on practical experience and a model that is good enough for the intended application.

LED diode: Recall the digital model of an LED as a switch

$$\text{diode switch} = \begin{cases} \text{closed (LED on)}, & V_{LED} \geq V_B, \\ \text{open (LED off)}, & V_{LED} < V_B \end{cases}$$



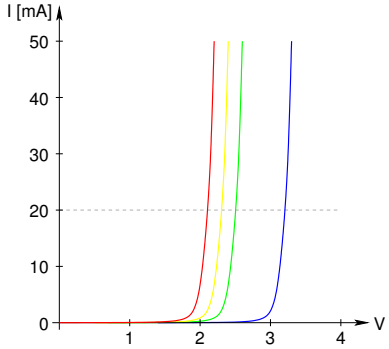
Models of Actuators

Q: Do LEDs with different colors behave like red LEDs?

A: No. Different colors originate from different materials that emit photons at different wavelengths, and affect the forward-bias voltage of the diode.

Typical LED characteristics:

Color	$\lambda[nm]$	V_{20mA}	Material
red	630–660	2.1V	GaAsP
yellow	580–590	2.3V	AlGaInP
green	550–570	2.5V	AlGaP
blue	450–500	3.2V	SiC

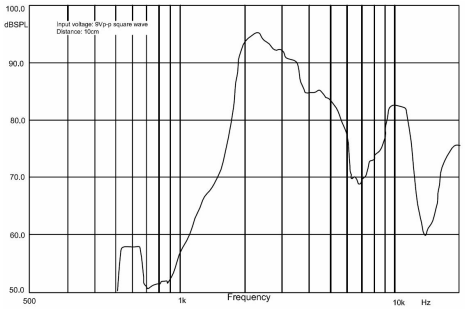


Note: Analog diode models have exponential IV-characteristics.

Models of Actuators

A piezoceramic element, piezo for short, functions as a loudspeaker or buzzer that generates air pressure.

We can model a piezo as a resonant RLC circuit. However, the resonant peak is not as distinguished in the measured frequency response as in an ideal RLC circuit: no model is perfect

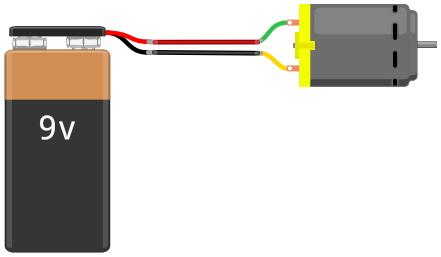


Practice: The resonant frequency of a buzzer is where it is loudest.

Models of Actuators

A DC motor is an actuator that converts electrical into mechanical energy and vice versa:

- Spin the shaft and you can measure a voltage at the terminals.
- Apply a voltage and the shaft spins.
- Apply the opposite voltage and the shaft spins in the opposite direction.



Models of Actuators

A DC motor can be modeled with the fundamental laws of physics:

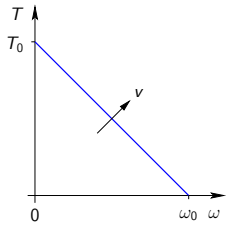
- The torque (rotational force) that acts on the shaft of a motor at rest is proportional to the current through the coils:

$$T(t) = k_T i(t)$$

- If the motor spins with angular velocity ω, the magnetic field induces a back electromotive force, back EMF for short, that opposes the rotation:

$$E_b(t) = k_e \omega(t)$$

- The maximum torque T_0 acts on the stalled shaft when $\omega = 0$.
- When the motor rotates at maximum velocity ω_0 then torque $T = 0$.



Models of Actuators

Assume the coil has resistance R and inductance L.

- The voltage at the coil terminals of a DC motor is:

$$v(t) = Ri(t) + L\frac{di(t)}{dt} + E_b$$

- For constant current $i(t) = T/k_t$ and $v(t) = V$, the motor spins at constant speed

$$\omega_m = \frac{V}{k_e} - \frac{RT}{k_e k_t}$$

- Let J be the moment of inertia of the shaft including load, then Newton's law ($F = ma$) for rotations is:

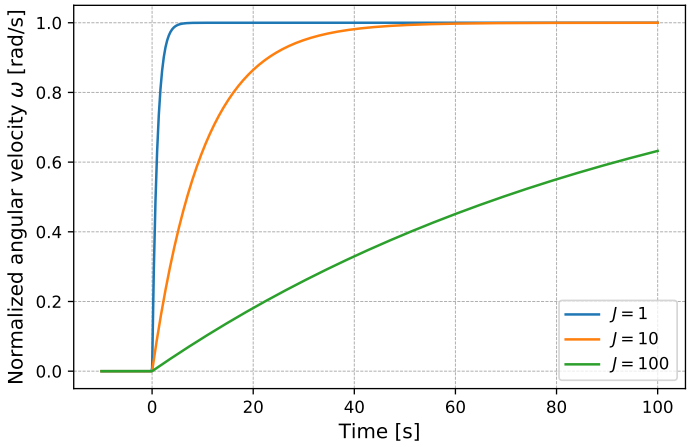
$$T(t) = J\frac{d\omega(t)}{dt}$$

- Applying a voltage V_0 to the motor at rest creates torque T_0, and the motor speed increases asymptotically

$$\omega(t) = \frac{k_T T_0}{k_e R}\left(1 - e^{-\frac{k_e R}{J k_T} t}\right)$$

Models of Actuators

Motor response to input voltage step $v(t) = V_0 u(t)$:



Models of Actuators

Q: How can we adjust the speed (RPM) of the motor?

A1: Relation $\omega_m(V)$: increase voltage V to increase ω_m.

Problem: Our model does not capture a second-order effect, friction.

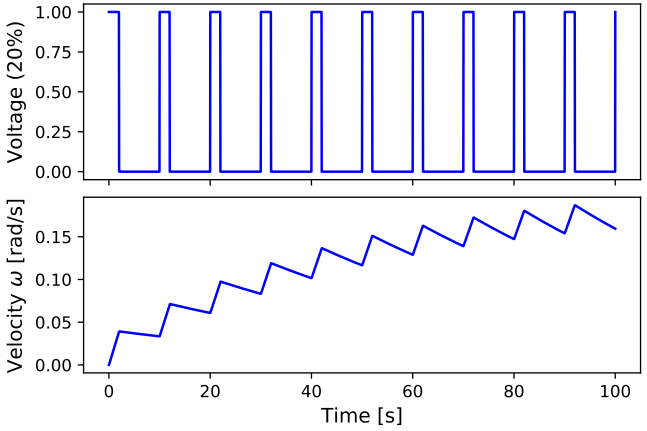
When the motor is at rest, friction prevents the motor from starting if the voltage is too low.

A2: Pulse width modulation (PWM) with nominal voltage.

The inertia of the motor and mechanical load suffices to smooth out abrupt changes of the voltage, e.g. when applying a square wave.

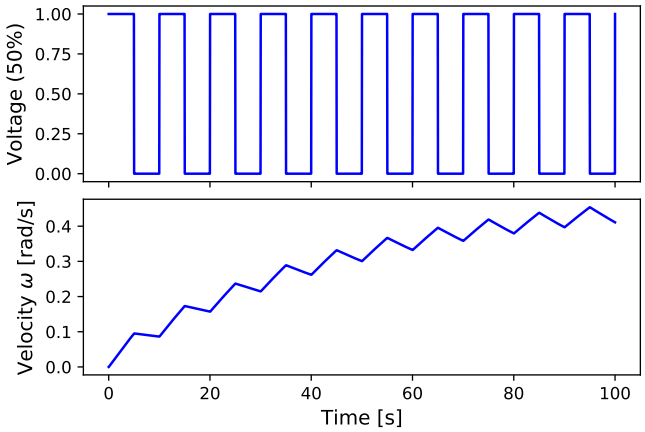
Models of Actuators

PWM enables us to control the motor speed. The larger the duty cycle, the faster the motor spins:



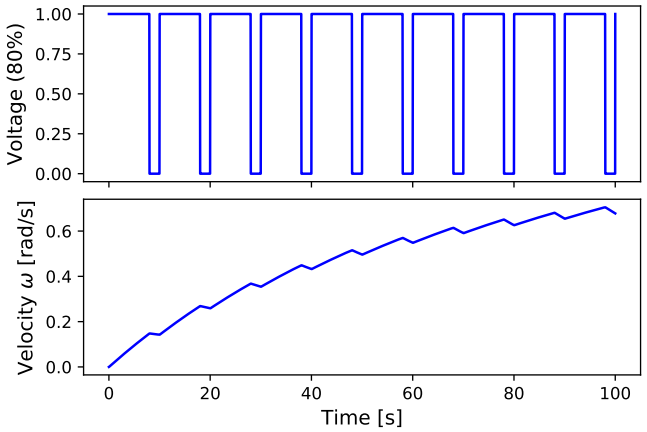
Models of Actuators

PWM enables us to control the motor speed. The larger the duty cycle, the faster the motor spins:



Models of Actuators

PWM enables us to control the motor speed. The larger the duty cycle, the faster the motor spins:

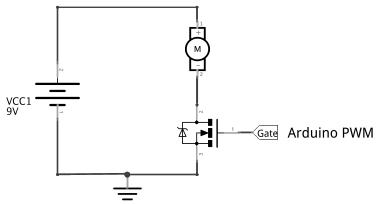


DC Motor Control

An MCU like an Arduino cannot drive a DC motor directly, because

- the DC motor requires a voltage of 9 V, whereas the maximum voltage on the Arduino Uno board is 5 V,
- the DC motor draws more current than an Arduino pin can supply.

We use a power MOSFET NPN transistor (IRF520) that can tolerate voltages up to $V_{DS} = 100\,\text{V}$ and currents up to $I_D = 9.7\,\text{A}$.



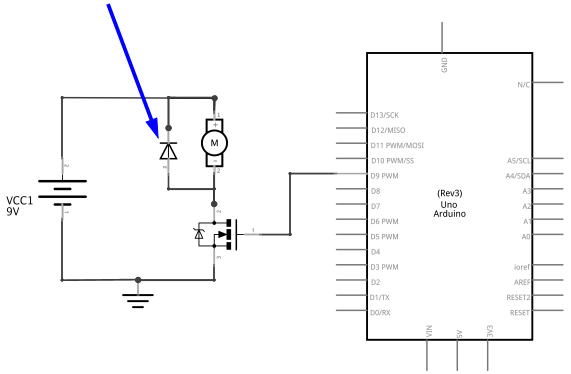
An Arduino pin can drive the gate of the transistor, i.e. a capacitor.

Operation: The Arduino PWM pin outputs a square wave that operates the transistor as a digital switch. The transistor isolates the Arduino from the high-voltage/high-current motor circuit.

DC Motor Control

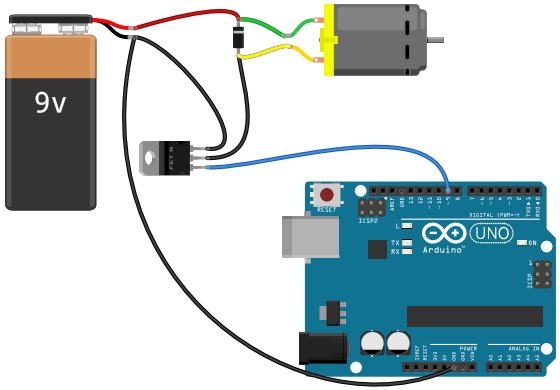
Problem: When a DC motor is switched off, the inductance continues to drive a current, that can build up a damaging voltage.

We add a flyback diode, aka snubber diode, in parallel to the motor to provide a path for dissipating the induced current.



DC Motor Control

Arduino circuit uses pin 9 for PWM output:



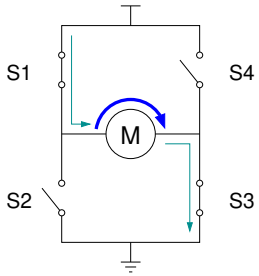
Careful: The power MOSFET can get very hot!

Arduino Sketch: DC Motor Speed Control

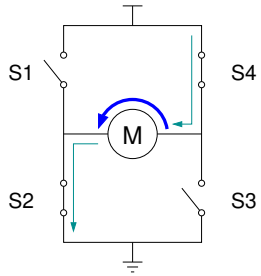
```
int pwmctrl = 128;  // must be in [0, 255] for analogWrite
void setup() { Serial.begin(9600); }
void loop() {
  int newpwm = pwmctrl;
  while (Serial.available() > 0) {
    int c = Serial.read();
    if (c == '+' && newpwm < 255)
      newpwm++;
    if (c == '-' && newpwm > 0)
      newpwm--;
  }
  if (newpwm != pwmctrl) {
    pwmctrl = newpwm;
    Serial.print("pwmctrl = ");
    Serial.println(pwmctrl);
    analogWrite(9, pwmctrl);    // adjust motor speed
  }
  delay(10);
}
```


DC Motor Control

To control both speed and direction of a motor without modifying the circuit, we need four switches:



Forward rotation

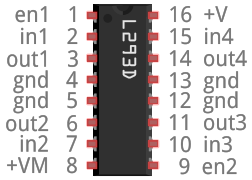


Backward rotation

This circuit is an H-Bridge. It is available as an IC with support for PWM speed control, and includes snubber diodes.

DC Motor Control

Pin assignment of H-bridge part L293D:



L293D features and connections:

- Two H-bridges for two motors. Apply a PWM signal to pin *en1* and/or *en2* for speed control of each motor.
- Pins *out1* and *out2* drive one DC motor directly. Pins *out3* and *out4* can drive a second motor.
- Supply 5 V to pin 16 to power the H-bridge, and 9 V to pin 8 which serves as power supply for the DC motors.

DC Motor Control

Motor control with L293D:

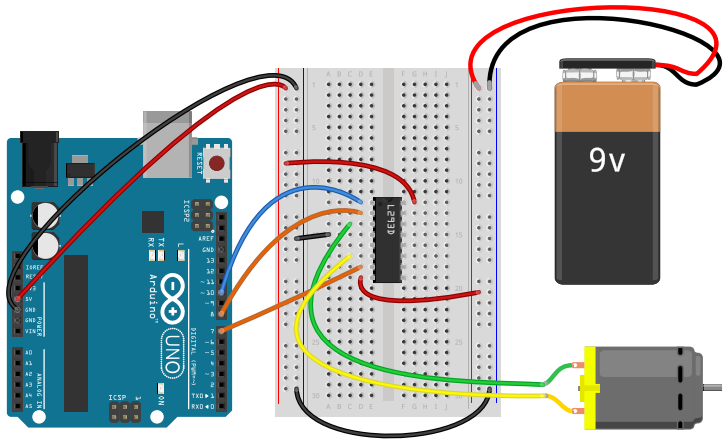
- The motor driven by pins *out1* and *out2* is speed controlled by PWM pin *en1*.
- The motor direction is selected via digital pins *in1* and *in2* according to the truth table:

in1	*in2*	Motor action
LOW	LOW	stop
HIGH	LOW	forward
LOW	HIGH	backward
HIGH	HIGH	stop

- A second motor can be speed controlled by PWM pin *en2*, and the direction is selected via digital pins *in3* and *in4*.

DC Motor Control

DC motor control with Arduino and H-bridge:



Arduino Sketch: DC Motor Control with H-Bridge

```
#define STOP  0   // motor directions
#define FWD   1
#define BWD   2

int dir = STOP;    // stop initially
int pwmctrl = 128; // must be in range [0, 255] for analogWrite

void setup() {
  Serial.begin(9600);
  analogWrite(10, pwmctrl);
}

void loop() {
  int newpwm = pwmctrl;
  int newdir = dir;

  while (Serial.available() > 0) {    // receive cmd from host
    int c = Serial.read();

    switch (c) {
    case 's':
      newdir = STOP;
      break;
```


Arduino Sketch: DC Motor Control with H-Bridge

```
    case 'f':
      newdir = FWD;
      break;
    case 'b':
      newdir = BWD;
      break;
    case '+':
      if (newpwm < 255)
        newpwm++;
      break;
    case '-':
      if (newpwm > 0)
        newpwm--;
      break;
    }
  }   /* end while*/

  if (newdir != dir) {                // set new direction
    switch (newdir) {
```

Arduino Sketch: DC Motor Control with H-Bridge

```
    case STOP:
      Serial.println("stop motor");
      digitalWrite(7, LOW);
      digitalWrite(8, LOW);
      break;
    case FWD:
      Serial.println("spin forward");
      digitalWrite(7, LOW);
      digitalWrite(8, HIGH);
      break;
    case BWD:
      Serial.println("spin backward");
      digitalWrite(7, HIGH);
      digitalWrite(8, LOW);
      break;
    }
    dir = newdir;
  }   /* endif set new direction */
```

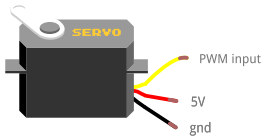

Arduino Sketch: DC Motor Control with H-Bridge

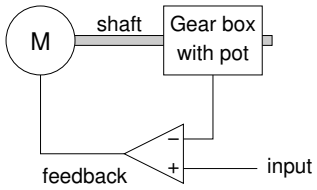
```
    if (newpwm != pwmctrl) {            // set new speed
      pwmctrl = newpwm;
      Serial.print("pwmctrl = ");
      Serial.println(pwmctrl);
      analogWrite(10, pwmctrl);         // adjust motor speed
    }

    delay(10);
  }
```

Careful: Stop the motor before reversing direction at high speed.

Servo Motor

A servo motor is a DC motor with an electrical feedback loop to control the position of the shaft with a PWM signal:

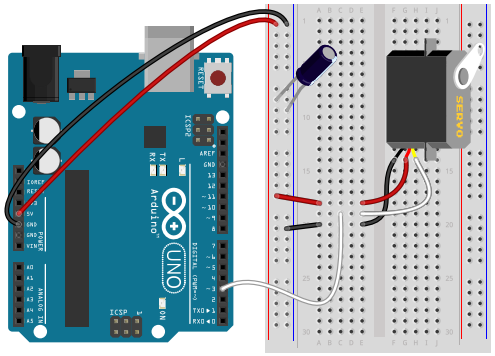




- Servo motors do not spin, but can lock the shaft into position.
- The gear box includes a mechanical position sensor, a cogwheel that turns a potentiometer.
- Affine model: The shaft position, i.e. the angle of the shaft, is proportional to the duty cycle of the PWM input, and the angle is limited to interval $[0°, 180°]$.

Servo Motor Control

Servo motor control with Arduino:



Decoupling capacitor (decap): The 100 μF electrolytic capacitor buffers current spikes when the servo turns on. Capacitive decoupling prevents voltage dips on the Arduino board that could otherwise result in malfunctioning of the MCU.

Arduino Sketch: Servo Motor Control

```
#include <Servo.h> // use Arduino Servo library
Servo servomotor;  // instantiate Servo object

int angle = 90;    // must be in range [0, 179] for Servo.write

void setup() {
  servomotor.attach(3);   // attach pwm pin 3
  Serial.begin(9600);
}

void loop() {
  int newangle = angle;

  while (Serial.available() > 0) {    // receive cmd from host
    int c = Serial.read();

    if (c == '+' && newangle < 179)
        newangle++;
    if (c == '-' && newangle > 0)
        newangle--;
  }
```


Arduino Sketch: Servo Motor Control

```
  if (newangle != angle) {
    angle = newangle;
    Serial.print("angle = ");
    Serial.println(angle);

    servomotor.write(angle);     // adjust angle
  }

  delay(10);
}
```