Arbeitskreis Hardware

Prof. Dr. Michael Rohs, Dipl.-Inform. Sven Kratz

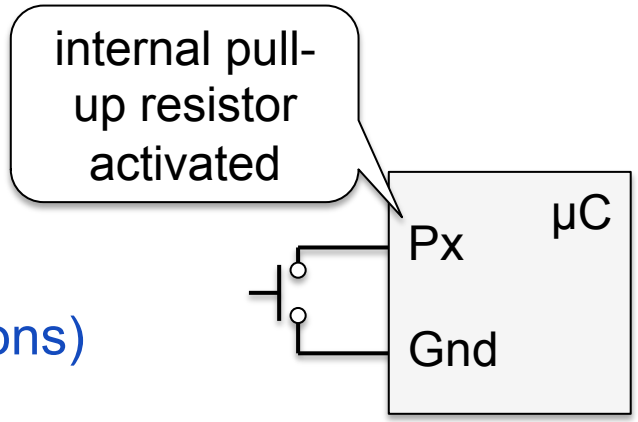
michael.rohs@ifi.lmu.de

MHCI Lab, LMU München

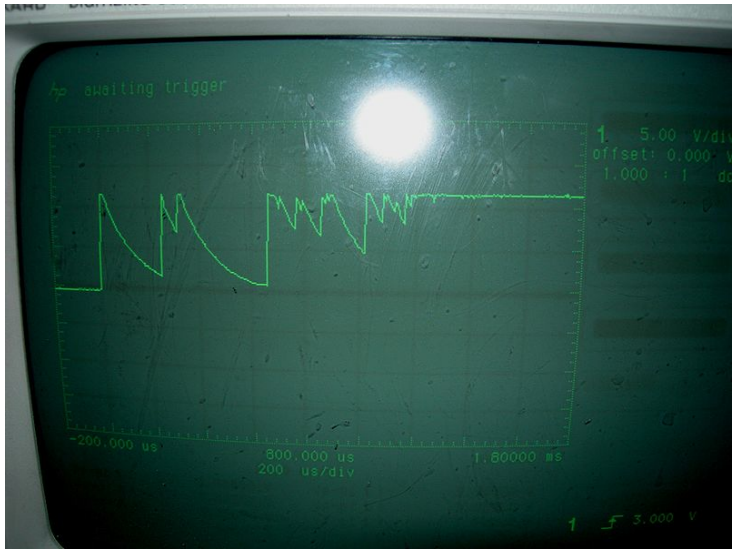
Schedule (preliminary)

Date	Topic (preliminary)
2.5.	Introduction to embedded interaction, microcontrollers, hardware & software tools
9.5.	*keine Veranstaltung (CHI)*
16.5.	soldering ISP adapter, AVR architecture
23.5.	LED displays, LED multiplexing, transistors, electronics basics
30.5.	AVR architecture, AVR assembler, sensors: light, force, capacity, acceleration, etc.
6.6.	PCB design & fabrication, EAGLE, 3D printing
13.6.	*keine Veranstaltung (Pfingsten)*
20.6.	I2C: interfacing to other chips (EEPROM, real-time clock, digital sensors)
27.6.	Displays (character LCDs, graphics LCDs), audio (speakers, amplification, op-amps)
4.7.	Actuation: stepper motors, servo motors
11.7.	Communication: fixed-frequency RF, ZigBee, Bluetooth
18.7.	Project
25.7.	Project

Button De-Bouncing



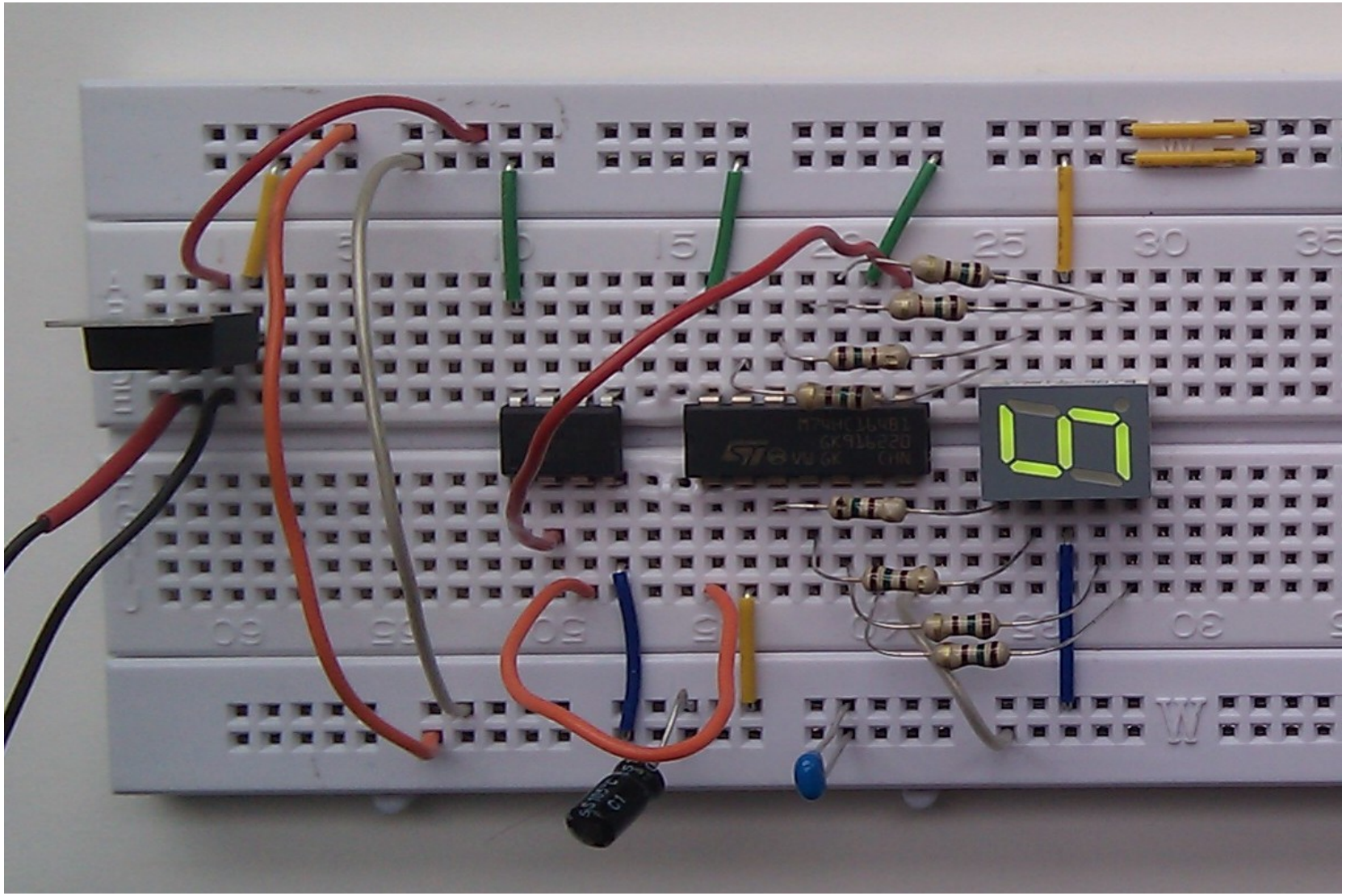
- Activate pull-up resistor on pin
  - Pull-up puts pin into defined state
  - (see previous slides on pin configurations)
- Connect button to GND
  - Pin will be high until button pressed
- De-Bouncing
  - Button contacts bounce, which generates many spikes
  - Hardware solutions: SR latch, capacitor
  - Software solution:
  - wait for 10-20ms after first event



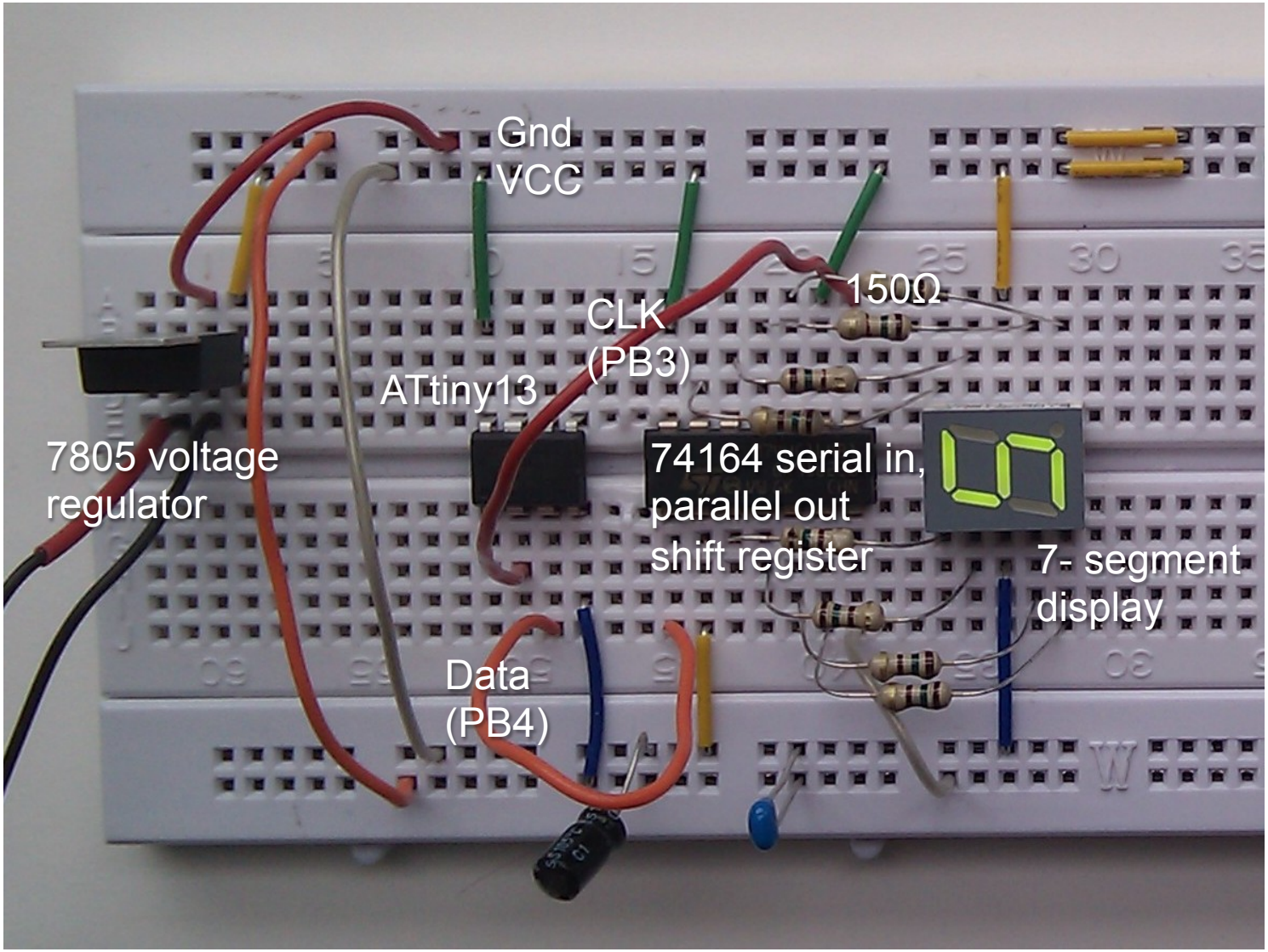
Source: Wikipedia, Author: Tomoldbury, public domain

LED DISPLAYS

7-Segment Display



7-Segment Display



7-Segment Display Code

```
#include <avr/io.h>
#include <util/delay.h>

//    76543210 (1 = on, 0 = off)
int digitPatterns[] = {
    0b11101110, // 0
    0b01100000, // 1
    0b11001101, // 2
    0b11101001, // 3
    0b01100011, // 4
    0b10101011, // 5
    0b10101111, // 6
    0b11100000, // 7
    0b11101111, // 8
    0b11101011, // 9
    0b00010000, // .
};
```

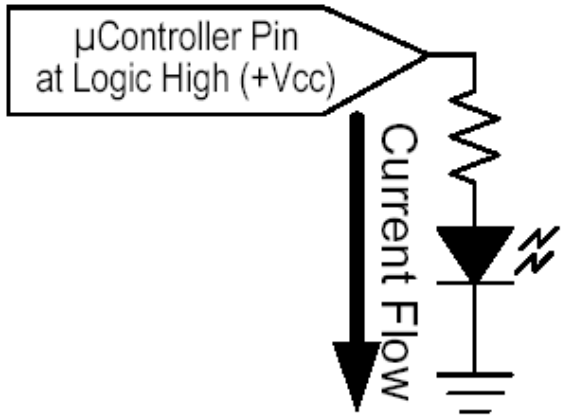
```
void pulseClock()
{
    // generate clock pulse
    // (port PB3 = clock)
    //        76543210
    PORTB |= 0b00001000;
    PORTB &= 0b11110111;
}
```

7-Segment Display Code

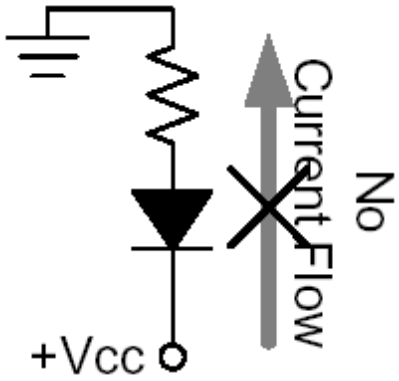
```
int main()
{
    // port PB3 = clock
    // port PB4 = data
    //                 76543210
    DDRB    |= 0b00011000; // PB3,4: output
    PORTB &= 0b11110111; // clock low

    // all segments off
    //                 76543210
    PORTB |= 0b00010000;
    int i;
    for (i = 0; i < 8; i++) {
        pulseClock();
    }
    _delay_ms(1000);

    i = 0;
    int j, d;

    while (1) {
        d = digitPatterns[i];
        for (j = 0; j < 8; j++) {
            if ((d & 0x80) == 0) {
                // off
                PORTB |= 0b00010000;
            } else {
                // on
                PORTB &= 0b11101111;
            }
            pulseClock();
            d <<= 1;
        }
        _delay_ms(1000);
        i++;
        if (i > 10) i = 0;
    }

    return 0;   /* never reached */
}
```

Using LEDs as Light Sensors

- LEDs are photo diodes
- Sensitive to light at the wavelength they emit
- Indirectly measure photo current
- Use LED under reverse bias conditions
  - Anode to ground
  - Cathode to +VCC



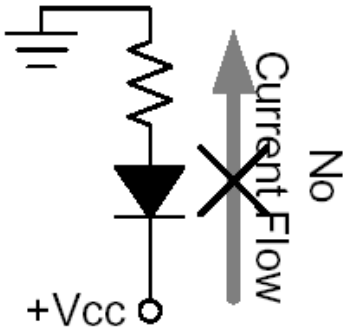
Normal operation



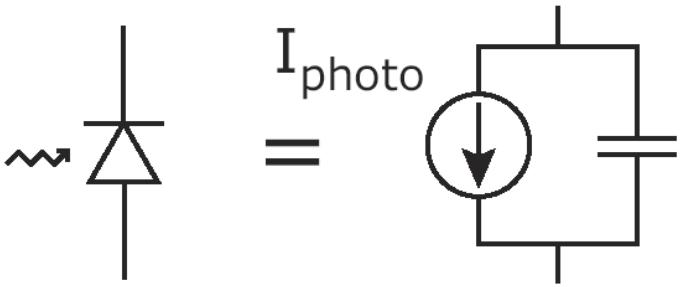
Reverse-biased LED

- Dietz, Yerazunis, Leigh: Very Low-Cost Sensing and Communication Using Bidirectional LEDs. UbiComp 2003, pp. 175-191.
- Hudson: Using Light Emitting Diode Arrays as Touch-Sensitive Input and Output Devices. UIST 2004, pp. 287-290.

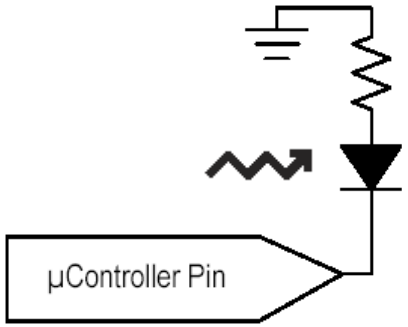
Using LEDs as Light Sensors



Reverse biased LED



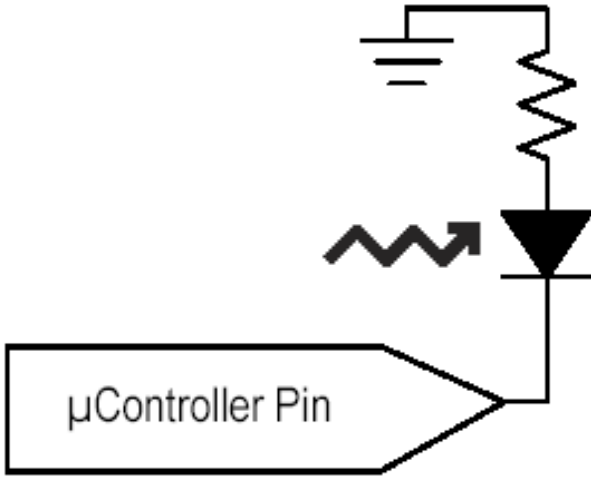
Modeled as capacitor in parallel with current source (photocurrent)



Circuit for light sensing

- Reverse biased LED = Capacitor parallel to photocurrent
- Sensing algorithm
  - Set microcontroller pin to high → capacitor charges
  - Switch pin to input mode → photo current discharges capacitor
  - Measure time until pin is low

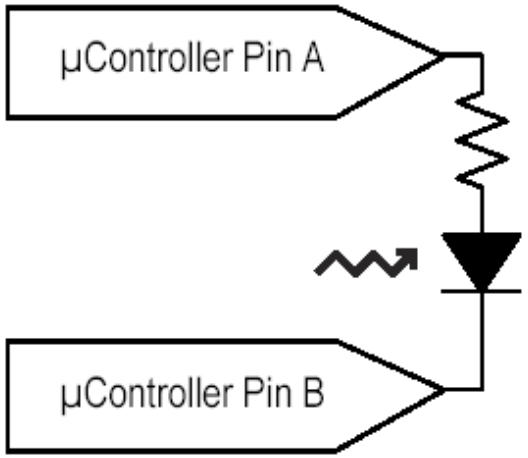
Using LEDs as Light Sensors




```
unsigned int senseLight() {
    // reverse bias LED
    DDRB  |= 0b00000001; // PB0 output
    PORTB |= 0b00000001; // PB0 high
    _delay_ms(10);

    // put in high-Z state
    DDRB  &= 0b11111110; // PB0 0 input
    PORTB &= 0b11111110; // PB0 low (turn off pull-up resistor)

    // count until pin drops to low
    unsigned int time = 0;
    while ((PINB & 1) == 1 && time < 0xffff) {
        time++;
    }
    return time;
}
```

Exercise: Using LEDs to alternate between Emitting and Sensing Light

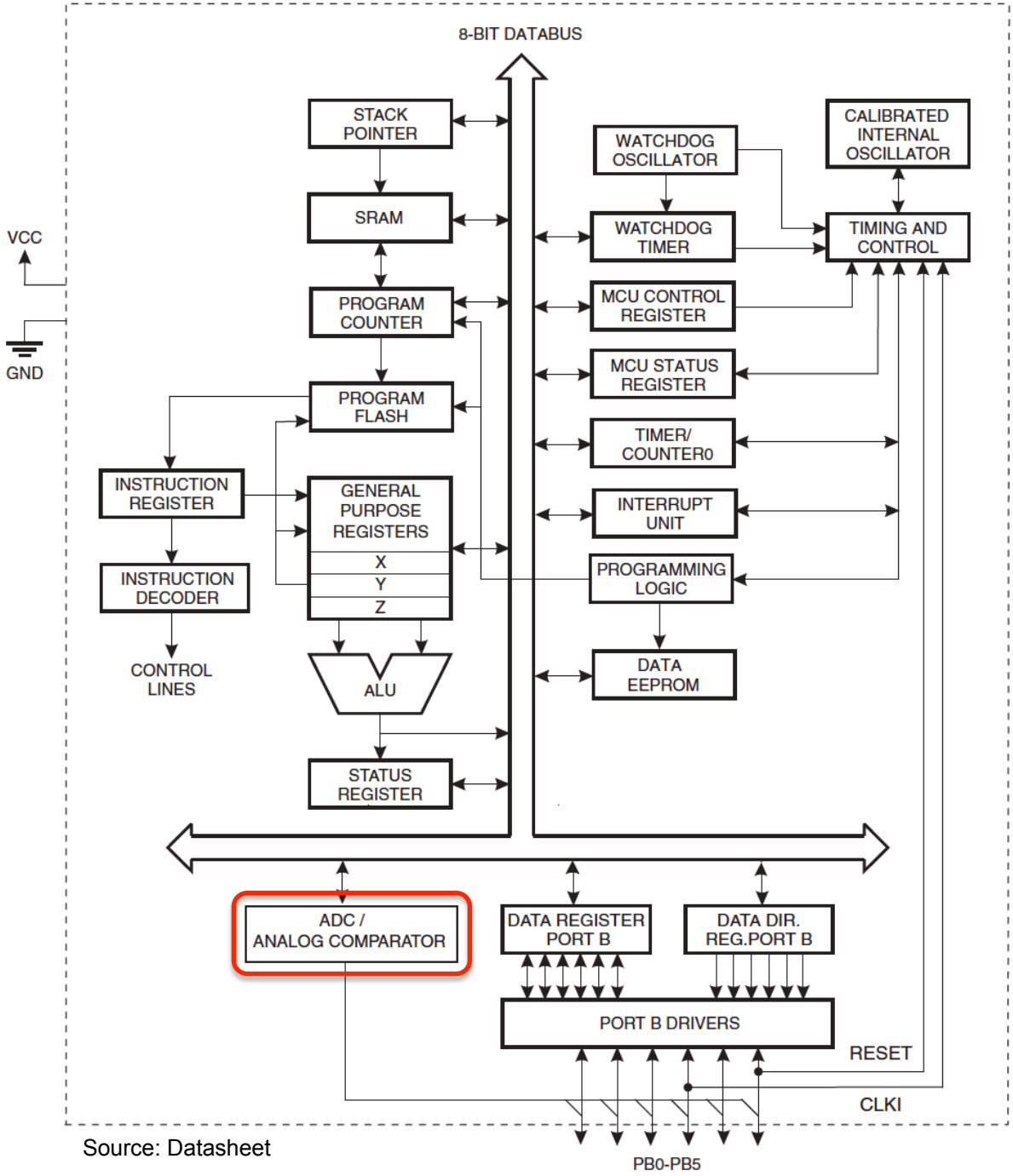
```
unsigned int senseAndEmitLight() {

   // …

    return time;
}
```



ANALOG DIGITAL CONVERSION

AVR ATtiny13 Architecture

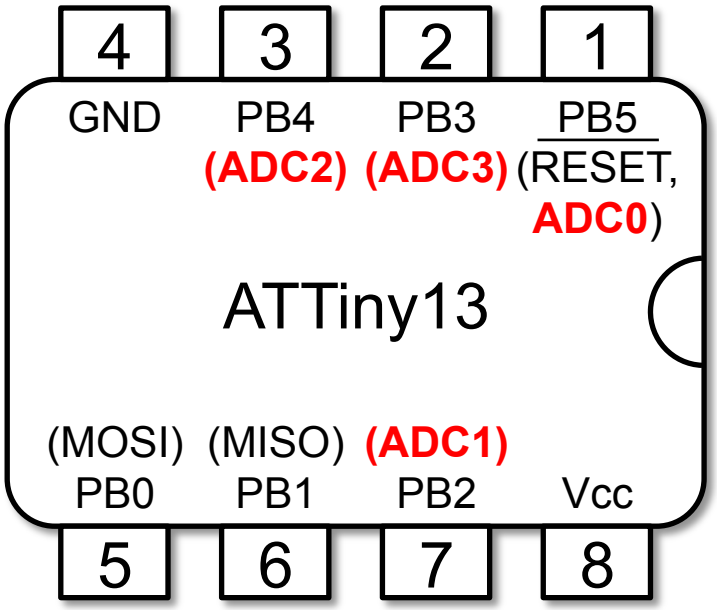
- Analog to Digital Converter



Source: Datasheet

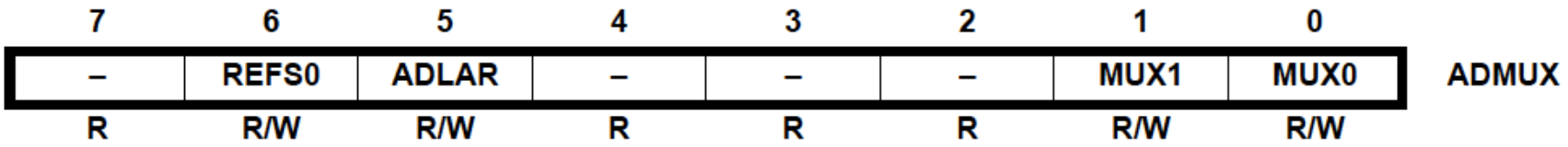
Analog to Digital Converter

- Converts analog input voltage to 10-bit digital value
  - value = 1024 * V_{IN} / V_{REF}
  - Min analog value = ground
  - Max analog value = V_{CC} or 1.1V internal voltage reference
  - 0x000 = ground
  - 0x3ff = ref. voltage - one LSB
- Characteristics
  - 4 input channels
  - 13-260µs conversion time
  - 15kSPS conversion rate
    (SPS = samples per second)



Analog to Digital Converter Registers

- ADMUX – ADC Multiplexer Selection Register



7	6	5	4	3	2	1	0	
–	REFS0	ADLAR	–	–	–	MUX1	MUX0	ADMUX
R	R/W	R/W	R	R	R	R/W	R/W	

Source: ATtiny13 Datasheet

- MUX0..1: Selecting input channel
  (00=ADC0, … , 11=ADC3)
- REFS0: Internal reference
  (0 = V_{CC} reference, 1 = internal 1.1V reference)
- ADLAR: ADC left adjust result
  (1 = left adjust result)

Analog to Digital Converter Registers

- ADCL and ADCH – The ADC Data Register, ADLAR = 0

15	14	13	12	11	10	9	8	
–	–	–	–	–	–	ADC9	ADC8	ADCH
ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
7	6	5	4	3	2	1	0	

Source: ATtiny13 Datasheet

- ADCL and ADCH – The ADC Data Register, ADLAR = 1

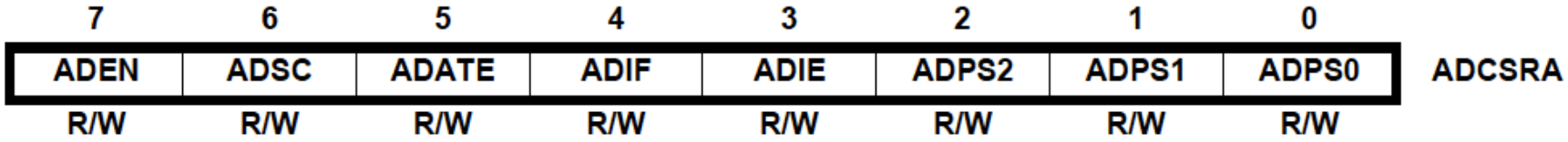
15	14	13	12	11	10	9	8	
ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
ADC1	ADC0	–	–	–	–	–	–	ADCL
7	6	5	4	3	2	1	0	

Source: ATtiny13 Datasheet

- If left adjusted: read ADCH to get 8-bit precision

Analog to Digital Converter Registers

- ADCSRA – ADC Control and Status Register A



7	6	5	4	3	2	1	0	
ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Source: ATtiny13 Datasheet

- ADEN: ADC enable (1 = on, 0 = off)
- ADSC: ADC start conversion, write 1 to start conversion
  - Reads 1 while conversion in progress
- ADATE: ADC auto trigger enable, write 1 to enable triggering on positive edge of trigger signal
- ADIF: ADC interrupt flag, set when conversion complete
- …

Analog to Digital Converter Registers

- ADCSRA – ADC Control and Status Register A

7	6	5	4	3	2	1	0	
ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

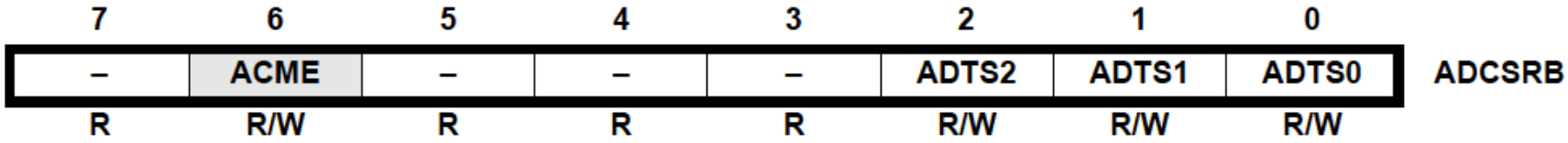
Source: ATtiny13 Datasheet

- …
- ADIE: ADC interrupt enable, write 1 (and set I-bit in SREG) to enable “ADC complete interrupt”
- ADPS2:0: ADC prescaler select bits, division factor of system clock to ADC clock

  0..7: 2, 2, 4, 8, 16, 32, 64, 128

Analog to Digital Converter Registers

- ADCSRB – ADC Control and Status Register B



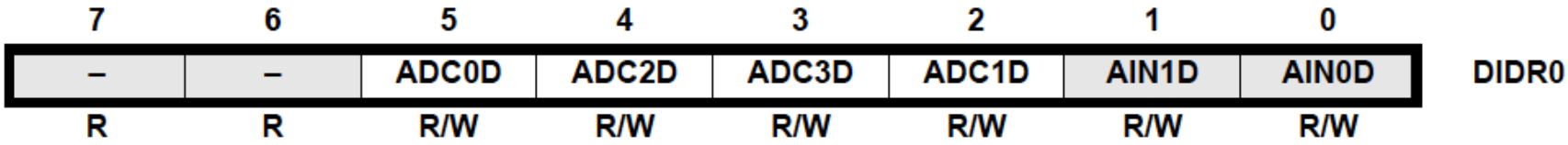
Source: ATtiny13 Datasheet

- ADTS2..0: ADC auto trigger source, triggers conversion on rising edge of selected interrupt

ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free Running mode
0	0	1	Analog Comparator
0	1	0	External Interrupt Request 0
0	1	1	Timer/Counter Compare Match A
1	0	0	Timer/Counter Overflow
1	0	1	Timer/Counter Compare Match B
1	1	0	Pin Change Interrupt Request

Analog to Digital Converter Registers

- DIDR0 – Digital Input Disable Register 0



7	6	5	4	3	2	1	0	
–	–	ADC0D	ADC2D	ADC3D	ADC1D	AIN1D	AIN0D	DIDR0
R	R	R/W	R/W	R/W	R/W	R/W	R/W	

Source: ATtiny13 Datasheet

- Write 1 to disable unneeded ADC input channels
- Reduces power consumption

Starting an Analog-to-Digital Conversion

- Starting a single conversion
  - Write 1 to start conversion bit (ADSC) in control register (ADCSRA)
  - ADSC reads 1 during conversion
- Auto triggering
  - Various sources (e.g. timer for fixed intervals)
  - Interrupt flag for source will be set and conversion triggered, even if interrupt is disabled, flag must be cleared to trigger subsequent conversion
  - Free-running-mode: start new conversion as soon as ongoing conversion has finished

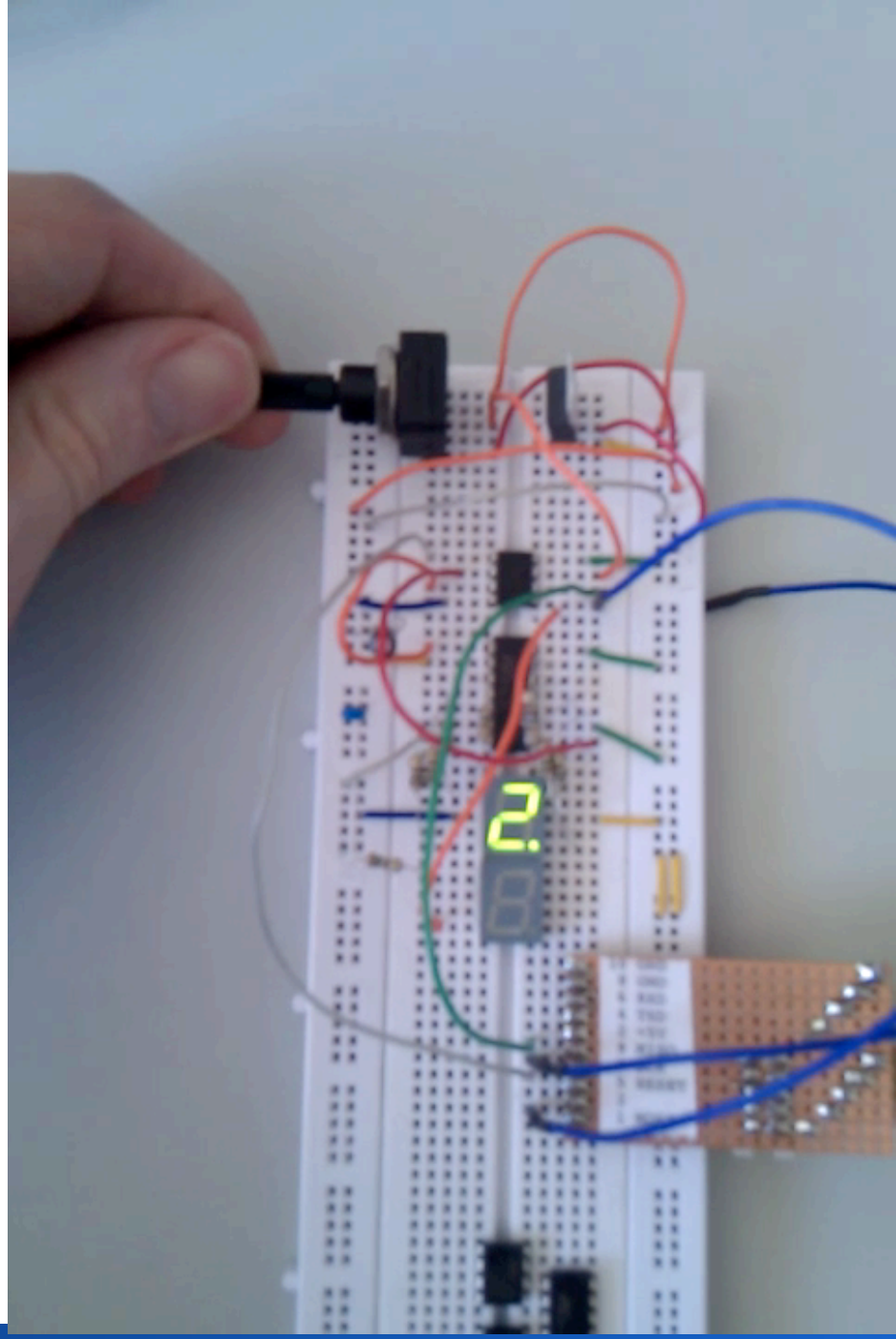
Conversion Timing

- Input clock frequency
  - 50..200 kHz ADC for maximum resolution
  - >200kHz if lower resolution sufficient
- Duration
  - Normal conversion takes 13 ADC clock cycles
  - Initial conversion 25 ADC clock cycles
- Discard first result after ADC activation or chg. ref.
  - Result may be wrong → discard

Analog Input Signal

- Optimized for impedance of 10kΩ or less
- Signal should be slowly varying
- Signal components higher than the Nyquist frequency (f_{ADC} / 2) should not be present
- Recommendations
  - Keep signal paths short
  - Separate analog tracks from digital tracks
  - Do not switch digital output pin during conversion
  - Place bypass capacitors close to V_{CC} and GND pins
  - Use ADC Noise Reduction Mode for higher accuracy

Example: 10kΩ on ADC1 (PB2)



Example: 10kΩ on ADC1 (PB2)

```
int main()
{
    DDRB  |= 0b00011000; // ports 3 and 4 output
    DDRB  &= 0b11111011; // port 2 as input
    PORTB &= 0b11110011; // port 3 (clock) low, port 2 disable pull-up

    ADMUX = (0 << REFS0) | (1 << ADLAR) | 1; // Vcc, left adjust, ADC1 = PB2
    ADCSRA = (1 << ADEN) | 6; // enable, prescaler 1:64 (150kHz)
    ADCSRB = 0; // free running mode
    DIDR0 = (1 << ADC0D) | (1 << ADC2D) | (1 << ADC3D);

    unsigned int d = 0;
    while (1) {
        ADCSRA |= (1 << ADSC);
        while (ADCSRA & (1 << ADSC)); // wait until conversion complete

        unsigned int l = ADCH & 0xff;
```

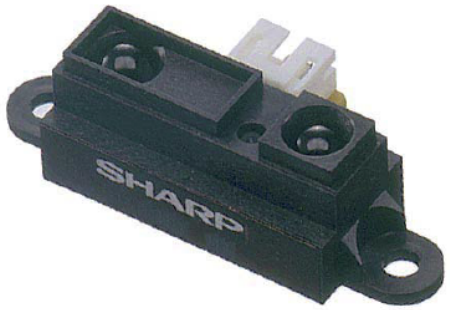
Example (ctd.)

```
    d = (l / 100) % 10;
        showDigit(d);
        showDigit(10);

        d = (l / 10) % 10;
        showDigit(d);
        showDigit(10);

        d = l % 10;
        showDigit(d);
        showDigit(10);

        showDigit(10);
        showDigit(11);
        showDigit(10);
        showDigit(11);
    }
    return 0;   /* never reached */
}
```

```
void showDigit(unsigned int d) {
    d = digitPatterns[d];
    int j;
    for (j = 0; j < 8; j++) {
        if ((d & 0x80) == 0) {
            // off: port PB4 (data) high (high = off)
            PORTB |= 0b00010000;
        } else {
            // on: port PB4 (data) high (low = on)
            PORTB &= 0b11101111;
        }
        pulseClock();
        d <<= 1;
    }
    _delay_ms(300);
}
```

SENSORS

IR Distance Sensors



- Sharp GP2Y0A21YK0F
  - Distance: 10 to 80 cm
  - Near infrared: λ = 870 nm
  - Operation: 5V, 30mA
  - Connectors: Vcc, GND, Vout
- Linearizing analog output

  distance = 27.93 * $V_{out}^{-1.2}$

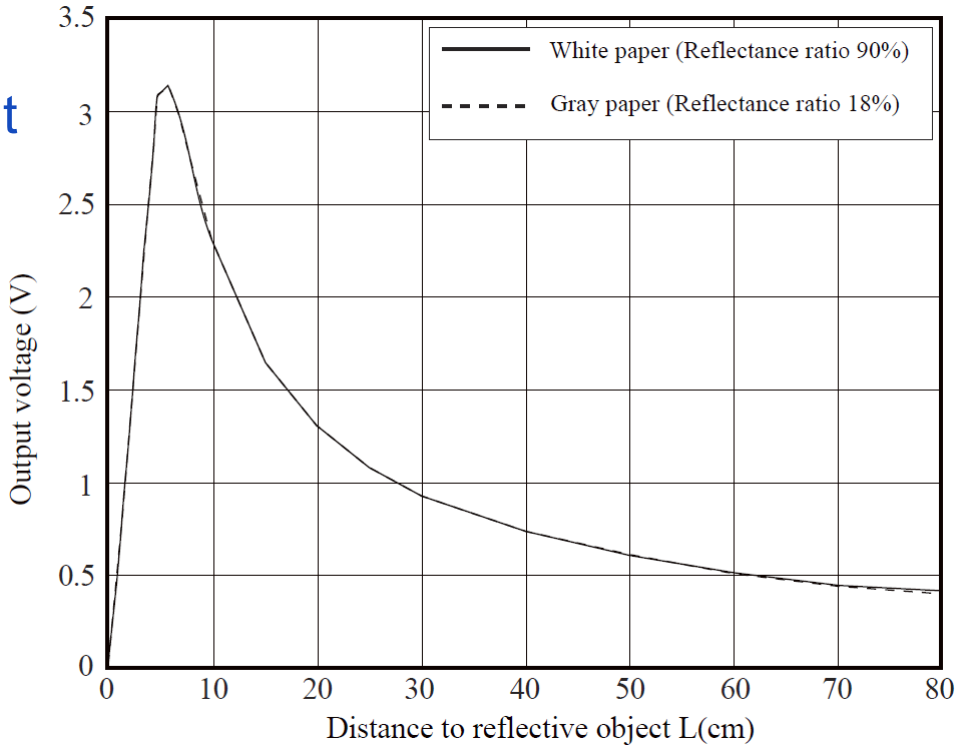
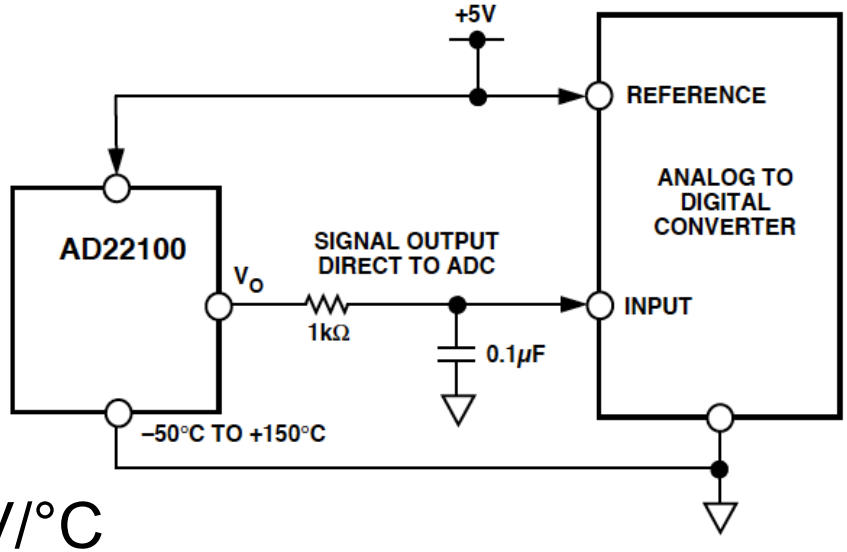
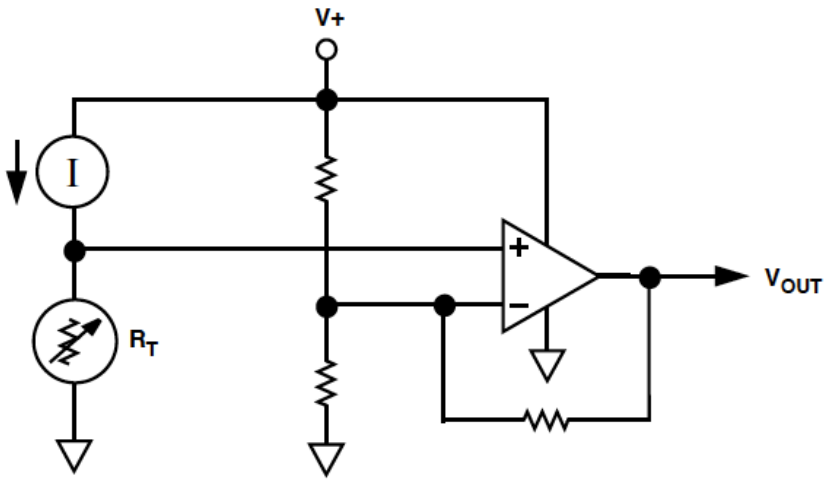


Figure sources: Sharp Datasheet

Temperature Sensor (Analog Devices AD22100)

- Temperature range: –50°C to +150°C
- Accuracy: < ±2%
- Linearity: < ±1%
- Output: V_{OUT} = 1.375 V + T * 0.0225 V/°C





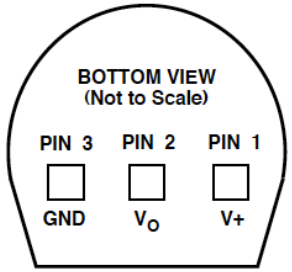
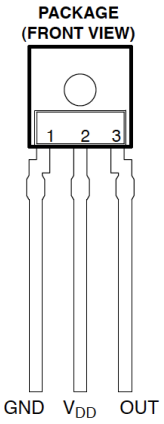
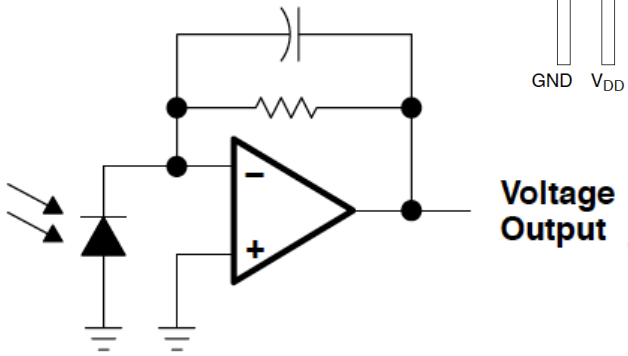


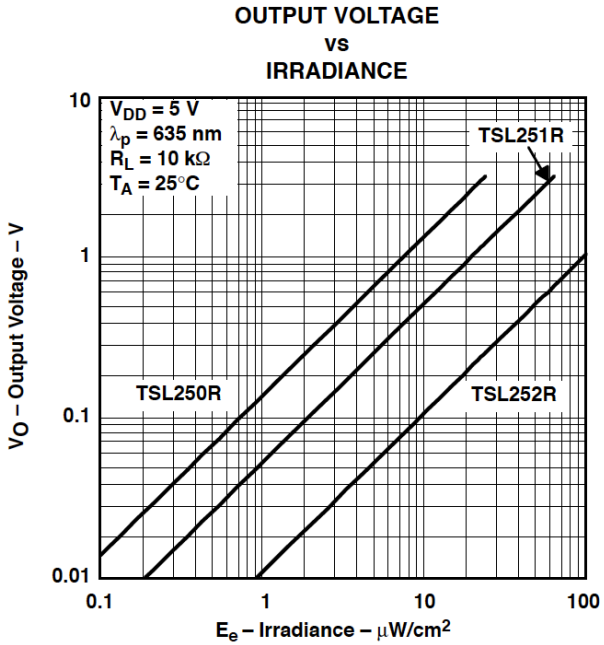
Figure sources: AnalogDevices Datasheet

Light-to-Voltage Sensors (TSL250R)

- Converts light intensity to voltage
- Photodiode + OpAmp
- Supply-voltage range: 2.7..5.5V
- Supply current: 1.1 mA







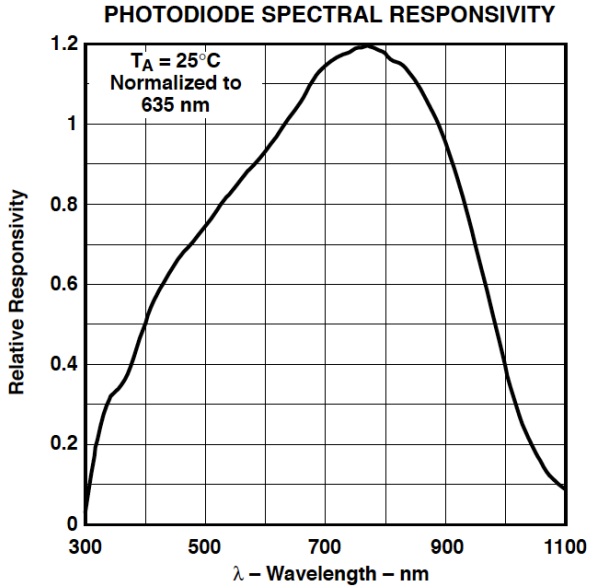
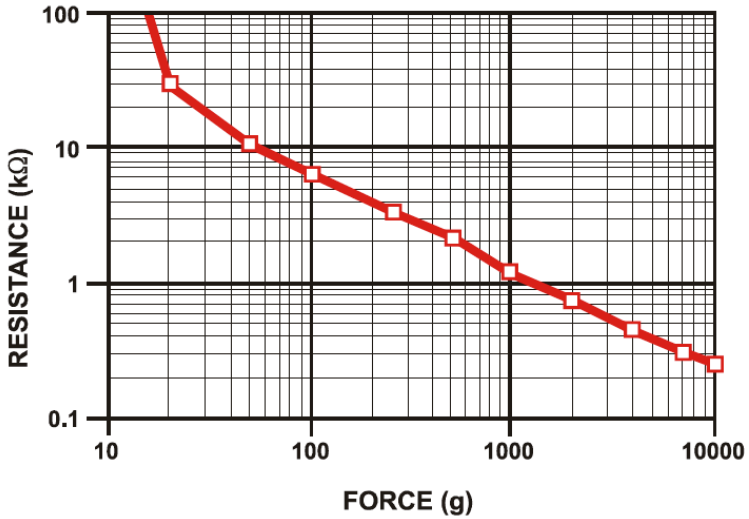


Figure sources: TAOS Datasheet

Pressure: Force Sensitive Resistors

- Composed of multiple layers
- Flat, sensitive to bend
- Force changes resistance



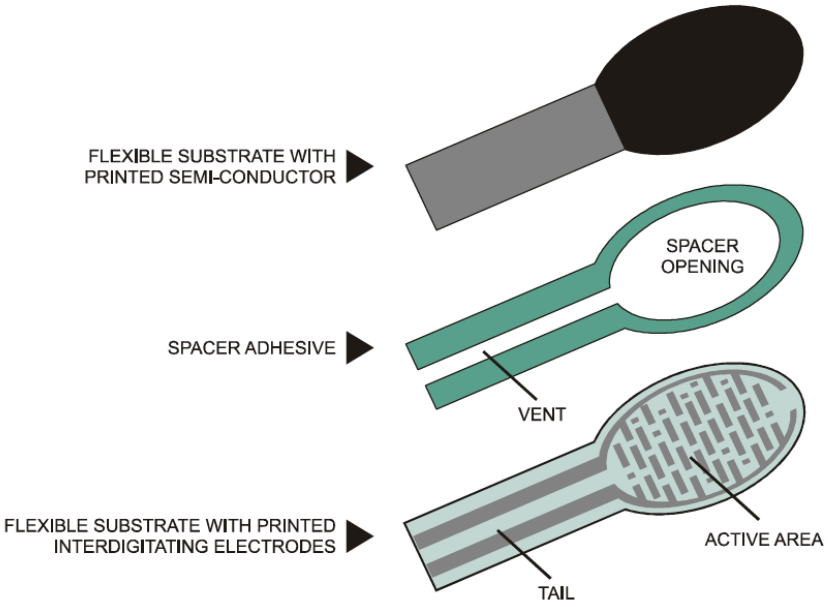
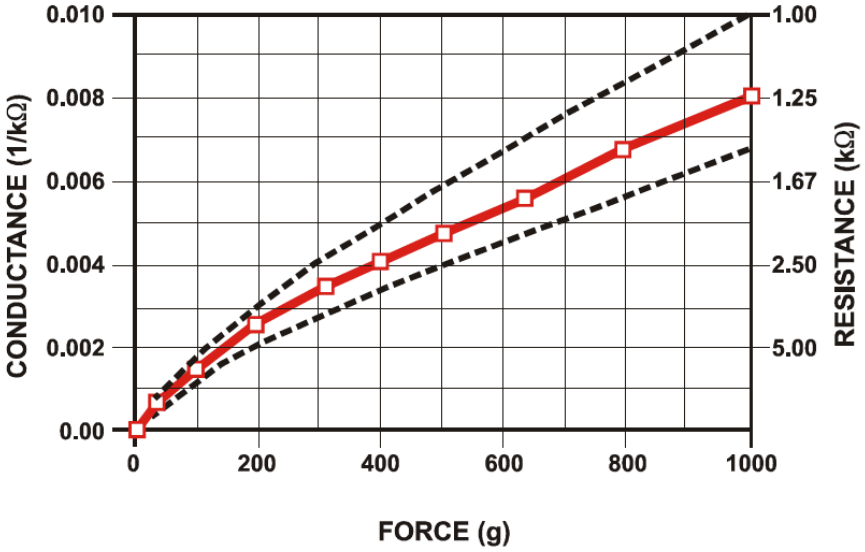


Figure sources: Interlink

- Non-linear response curve
- Sensed value depends on physical coupling → fingers

FSR Characteristics

- Low force range 0..1kg important for human interaction
- Not very precise: force accuracy 5..25%
- But humans are even worse in judging pressure



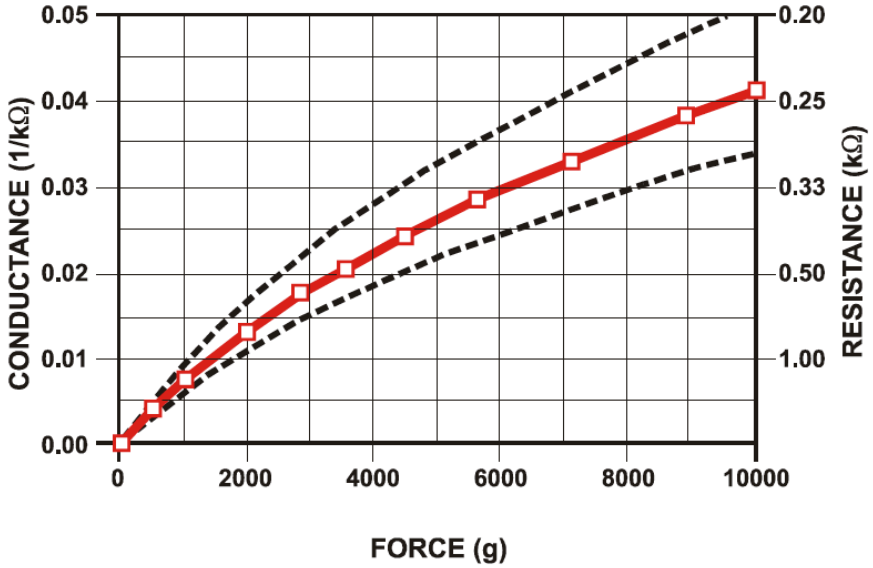
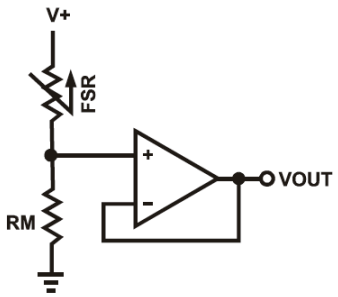
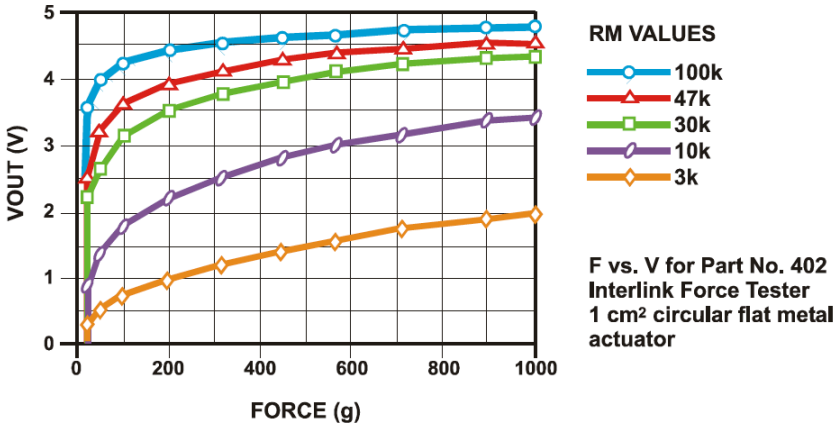


Figure sources: Interlink

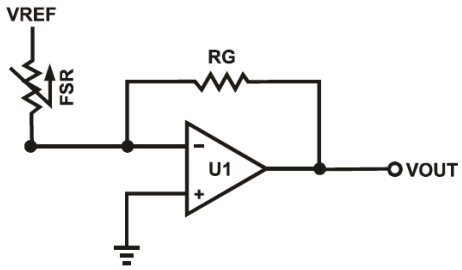
FSR Interface Circuits

- Voltage divider
  - Very nonlinear
  - Behaves like switch





- Current-to-voltage converter
  - Better dynamic range



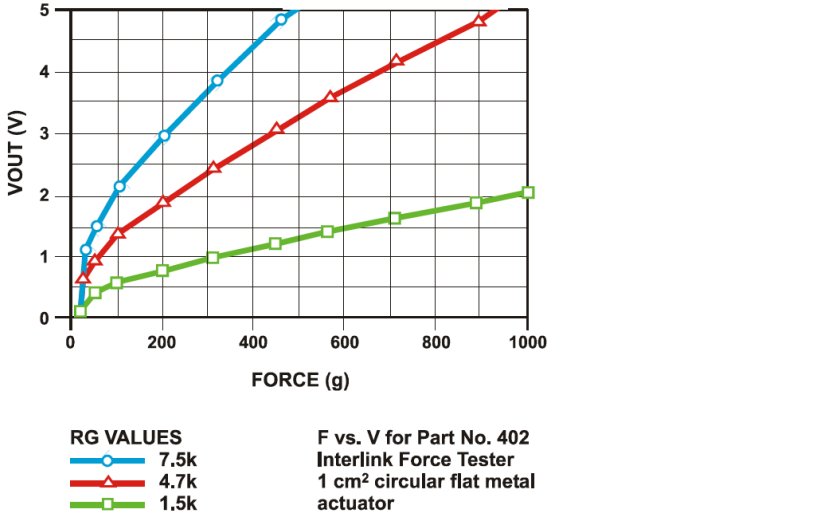
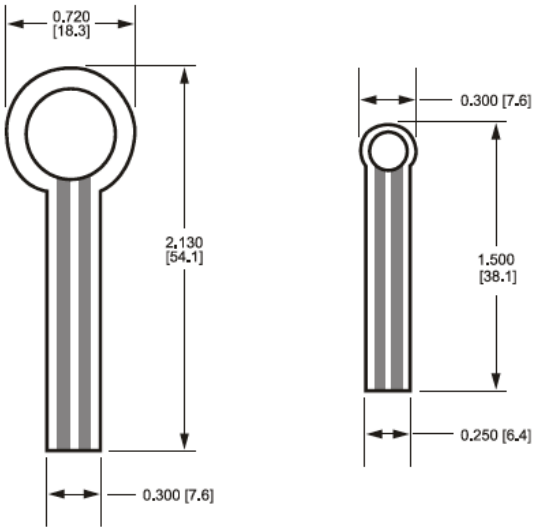
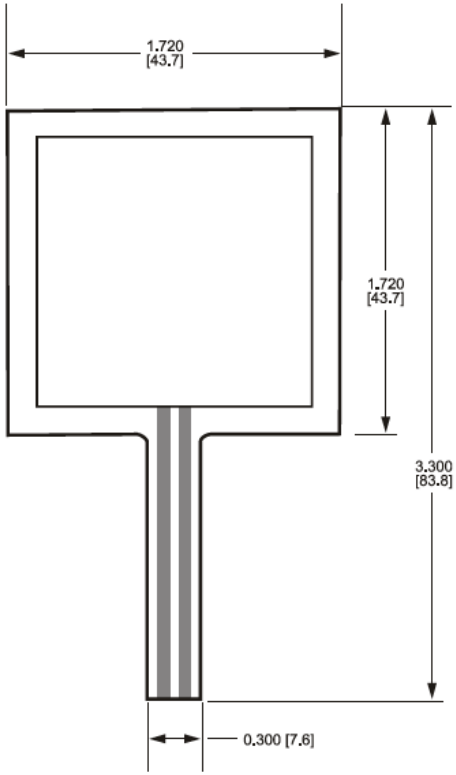


Figure sources: Interlink

FSR Shapes

- Examples from Interlink





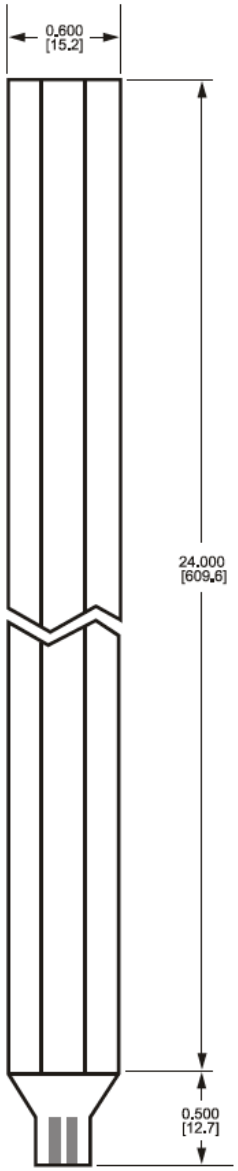
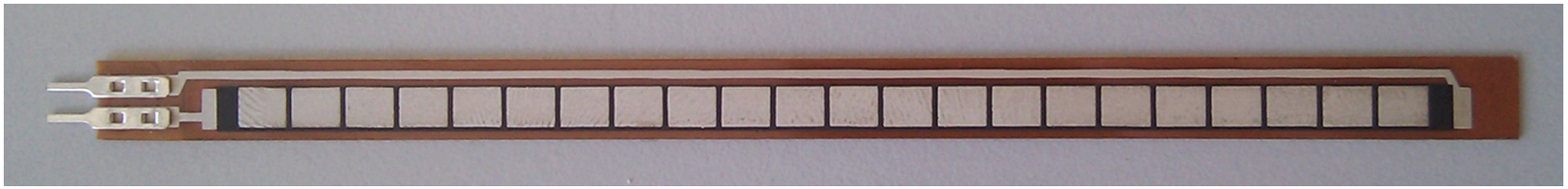


Figure sources: Interlink

FSR Bend Sensors

- Change resistance when bent
- Un-flexed: 10kΩ Flexed / bent 90°: 30..40kΩ

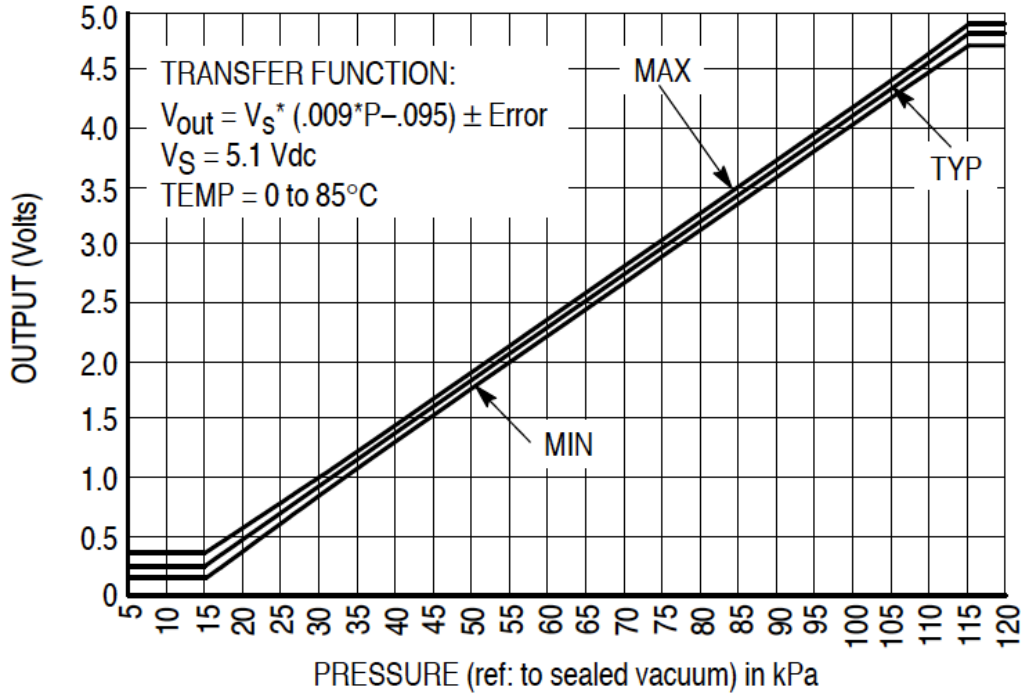


- Sensor glove
  - http://www.tufts.edu/programs/mma/emid/projectreportsS04/moerlein.html

http://www.tufts.edu/programs/mma/emid/projectreportsS04/moerlein.html

Air Pressure Sensors (MPX 4115A)

- Senses absolute air pressure in altimeter or barometer applications
- High level analog output signal
- Temperature compensation



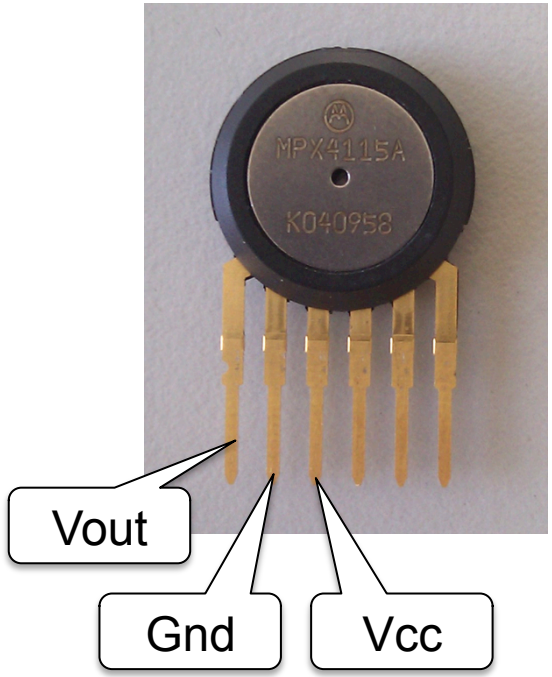
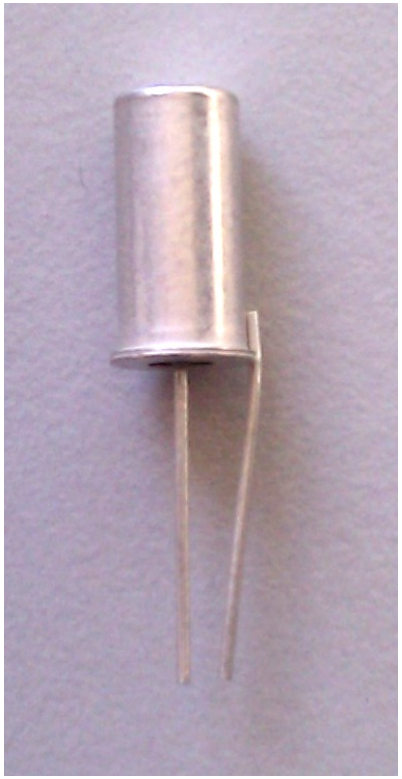


Figure sources: Motorola Datasheet

Tilt Switches

- Contains ball contact
- Contact closes based on gravity

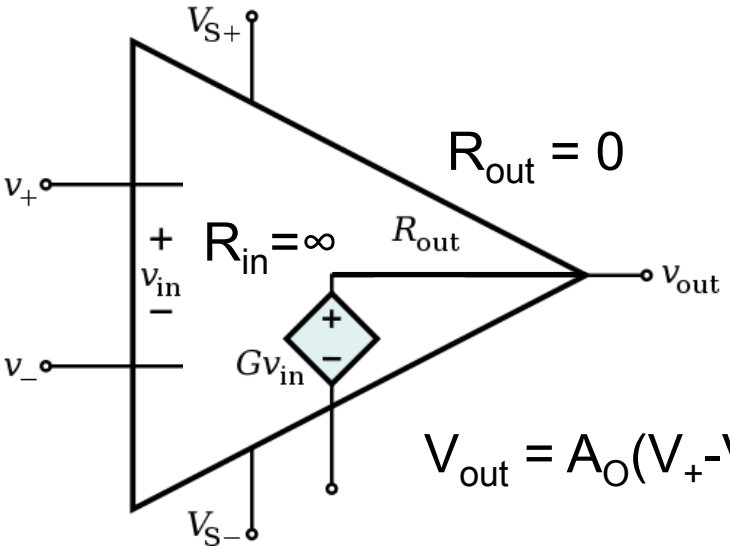


OPERATION AMPLIFIERS

Operational Amplifiers

- Ideal op-amps
  - Rule 1: open-loop voltage gain: $A_O = \infty$
  - Rule 2: Infinite input impedance: $R_{in} = \infty$
  - Rule 3: Input terminals draw no current

Ideal op-amp:



Real op-amp:

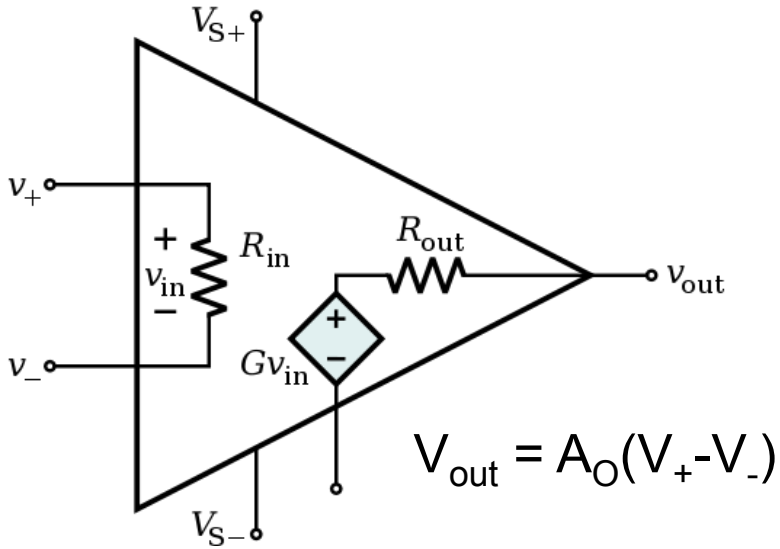
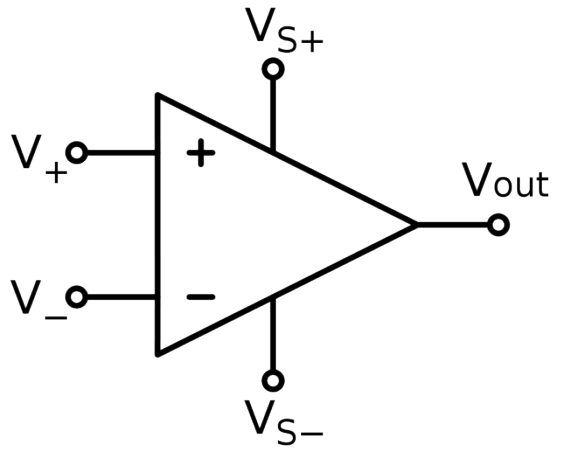
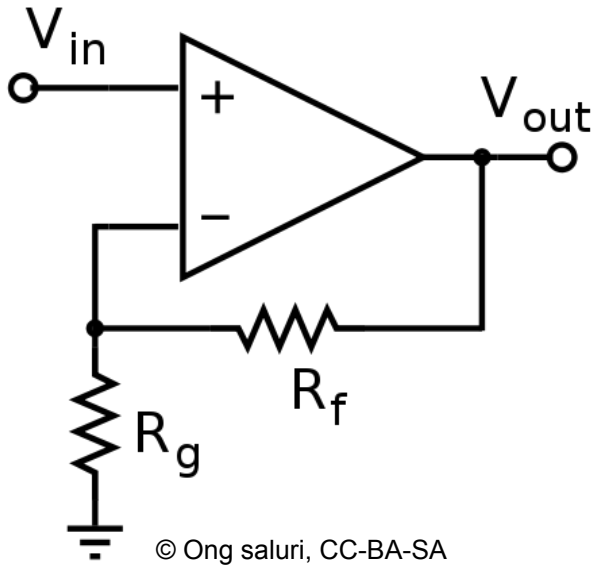


Figure author: Inductiveload, public domain

Operational Amplifiers

- Op-amp circuit diagram
  - V_+: non-inverting input
  - V_-: inverting input
  - V_{out}: output
  - V_{S+}: positive power supply
  - V_{S-}: negative power supply
- Negative feedback
  - Rule 4: $V_+ - V_- = 0$
  - Closed-loop operation
  - Used to control gain





Inverting Amplifier

- $V_+ = V_- = 0V$
- $I = V_{in} / R_{in}$
- $-I = V_{out} / R_f$
- Gain:
  $V_{out} / V_{in} = - R_f / R_{in}$

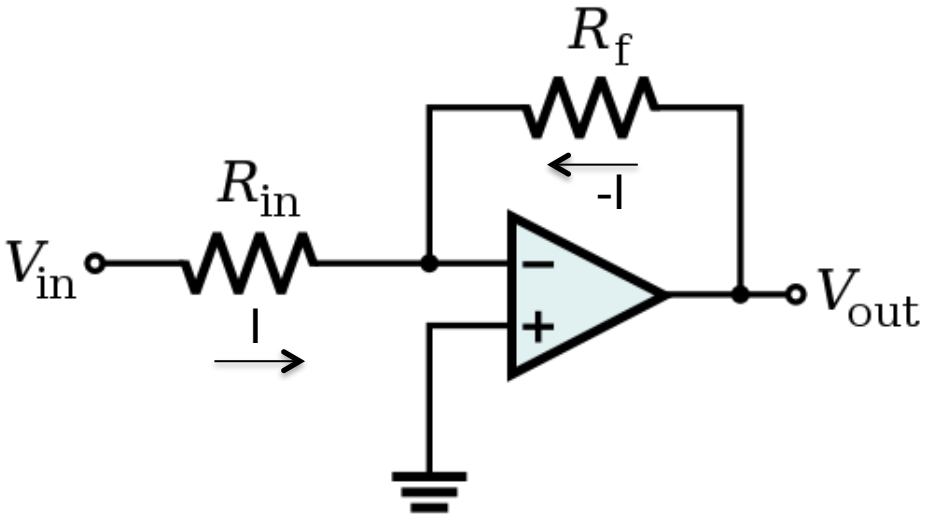


Figure author: Inductiveload, public domain

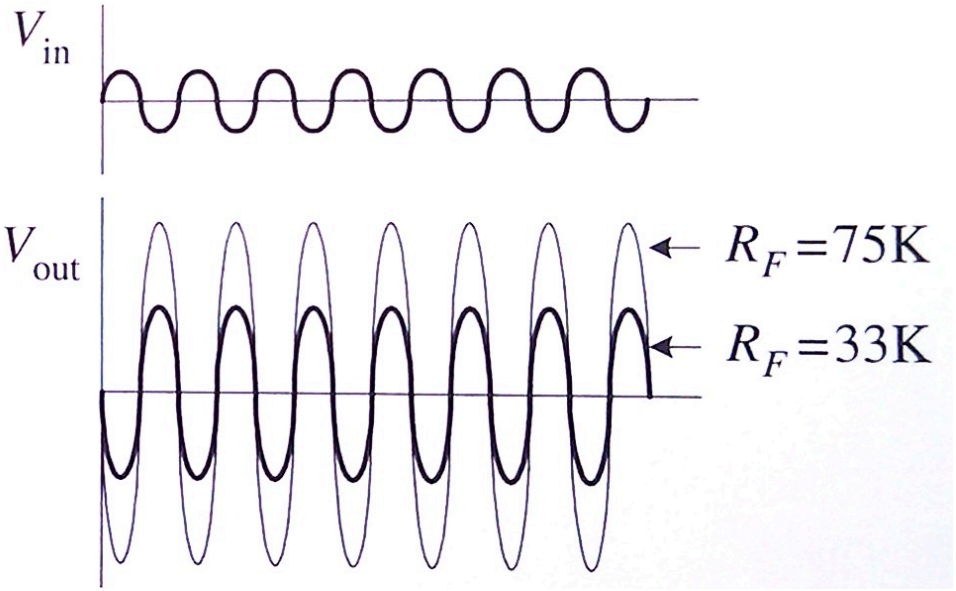


Figure: Paul Scherz: Practial Electronics for Inventors. 2nd edition, McGraw-Hill, 2007.

Non-Inverting Amplifier

- $V_{in} = V_+ = V_- = R_1 / (R_1 + R_2) * V_{out}$
- Gain:
  $V_{out} / V_{in} = (R_1 + R_2) / R_1 = 1 + R_2 / R_1$

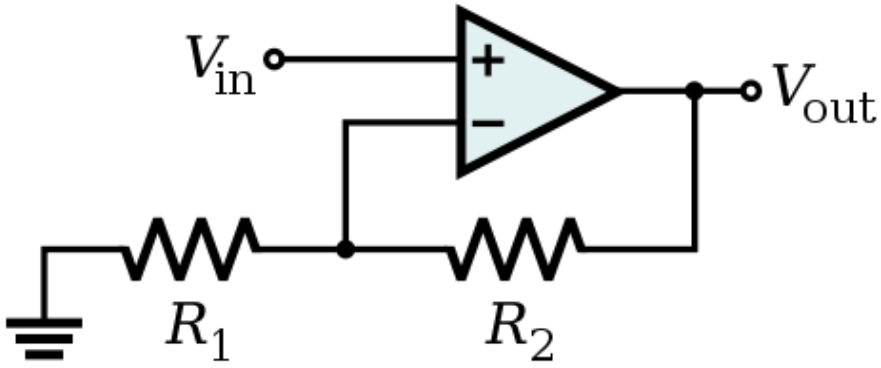


Figure author: Inductiveload, public domain

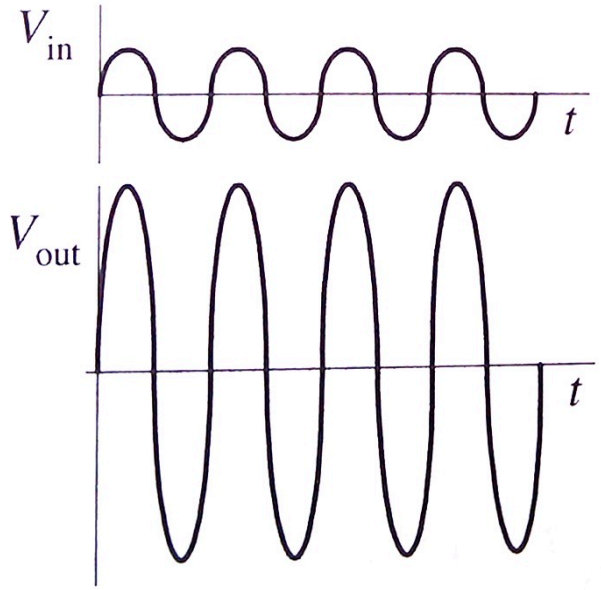


Figure: Paul Scherz: Practial Electronics for Inventors. 2nd edition, McGraw-Hill, 2007.

Unity Gain Buffer

- Gain: $V_{out} / V_{in} = 1$
- Benefits
  - high input impedance
  - low output impedance

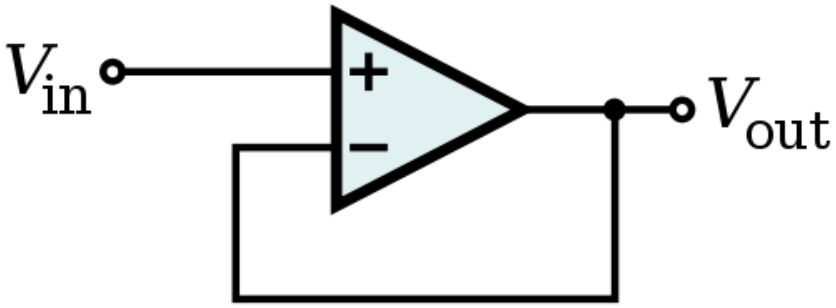


Figure author: Inductiveload, public domain

- High input impedance
  - does not draw much current from source
- Low output impedance
  - drives load like a good voltage source

- Put resistor on feedback loop to match source impedance
  - minimizes error due to bias current