Embedded Systems

Prof. Dr. Volker Strumpen

Rhine-Waal University of Applied Sciences

Winter Semester 2019

Syllabus

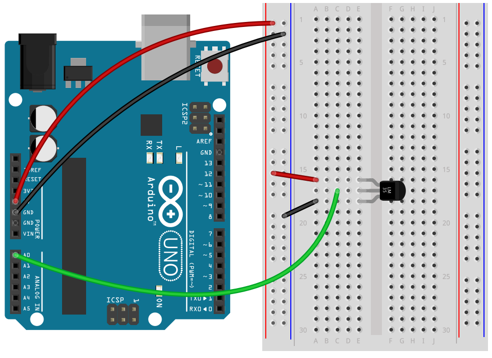
No	Date	Topic
1	10/2	Introduction
2	10/9	Facing the World
3	10/16	Sensors
4	10/23	Actuators
5	10/30	AVR Architecture
6	11/6	MCU System Architecture
7	11/13	C Programming
8	11/20	Assembly Programming
9	11/27	Timers and Interrupts
10	12/4	Communication
11	12/11	Real-Time Systems
12	12/18	Scheduling
13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

Outline

- Models of Sensors
- Dynamic Range
- Calibration

Temperature Sensor

Recall: The TMP36 temperature sensor converts analog temperature into analog voltage.



Temperature Sensor

Arduino sketch: Sample ADC output of analog voltage at pin A0.

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int temp = analogRead(A0);    // sample ADC output
  Serial.print("Temp: ");       // send value to host
  Serial.println(temp);

  delay(1000);                  // wait 1s
}
```

Q: How to associate value `temp` with actual temperature?

A: Devise a model of the temperature sensor.

Models of Sensors

A sensor measures a physical quantity $x(t)$, e.g. temperature.

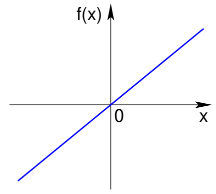
The sensor reports quantity $f(x(t))$, often a voltage.

Function $f\colon \mathbb{R} \to \mathbb{R}$ maps a real physical quantity $x(t)$ to a real physical quantity $f(x(t))$.

A linear function $f\colon \mathbb{R} \to \mathbb{R}$ has the form

$$f(x) \;=\; ax$$

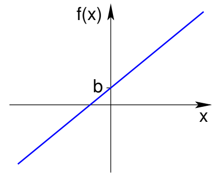
for some constant $a \in \mathbb{R}$.



If the straight line $f(x)$ does not pass through the origin, i.e. $f(0) \neq 0$, then

$$f(x) = ax + b\,, \quad b \neq 0$$

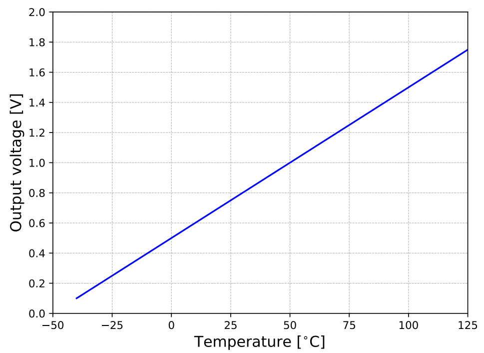
is an affine function with bias $b \in \mathbb{R}$.



Models of Sensors

Example: From datasheet of temperature sensor TMP36

Temperature range:	-40 °C to 150 °C / -40 °F to 302 °F
Output range:	0.1 V (-40 °C) to 2.0 V (150 °C), but accuracy decreases above 125 °C
Power supply:	2.7 V to 5.5 V



Models of Sensors

Affine model: $f(x(t)) = ax(t) + b$:

$$\begin{aligned} a &= \frac{2 - 0.1}{150 - (-40)} \;=\; 0.01\,\frac{\text{V}}{{}^\circ\text{C}} \\ b &= f(-40) - a(-40) \;=\; 0.1 + 0.01 \cdot 40 \;=\; 0.5\,\text{V} \\ f(x(t)) &= (0.01x(t) + 0.5)\,\text{V} \;=\; (10x(t) + 500)\,\text{mV} \end{aligned}$$

assuming temperature value $x(t)$ has unit °C.

Let voltage $v = f(x)$ and temperature $T = x$, then

$$v \;=\; 0.01T + 0.5$$

TMP36 model of temperature in °C as a function of voltage v in V:

$$T \;=\; 100v - 50$$

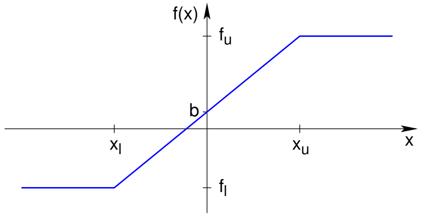
Models of Sensors

The operating range of a sensor specifies the physical values x that the sensor can measure. The range is always limited to an interval:

$$x_l \leq x \leq x_u .$$

For values $x < x_l$ and $x > x_u$ sensors saturate, s.t.

$$f(x) = \begin{cases} ax_l + b = f_l\,, & x < x_l\,, \\ ax + b\,, & x_l \leq x \leq x_u\,, \\ ax_u + b = f_u\,, & x > x_u \end{cases}$$



Models of Sensors

Example: Temperature sensor TMP36 has temperature range

$$T_l = -40\,^\circ\mathrm{C} \leq T \leq T_u = 150\,^\circ\mathrm{C}$$

and limits the output voltage to

$$v_l = 0.1\,\mathrm{V} \leq v \leq v_u = 2.0\,\mathrm{V}$$

Updated TMP36 model for output voltage v in V:

$$v = \begin{cases} 0.1\,, & T < -40\,, \\ 0.01\,T + 0.5\,, & -40 \leq T \leq 150\,, \\ 2.0\,, & T > 150 \end{cases}$$

If the accuracy above $125\,^\circ$C is unacceptably low, we may limit the operating range to $T_u = 125\,^\circ$C in software, s.t.

$$v = \begin{cases} 0.1\,, & T < -40\,, \\ 0.01\,T + 0.5\,, & -40 \leq T \leq 125\,, \\ 1.75\,, & T > 125 \end{cases}$$

Models of Sensors

Output voltage $v(t)$ of TMP36 is connected via pin A0 to the input of the ADC:

$$T(t) \longrightarrow \boxed{\text{TMP36}} \xrightarrow{v(t)} \boxed{\text{ADC}} \longrightarrow \texttt{analogRead()}$$

The ATmega328P ADC quantizes analog input voltage V_{in} in range

$$0\,\mathrm{V} \leq V_{in} < 5\,\mathrm{V} = V_{ref}$$

producing a 10-bit digital output d in interval

$$0 \leq d < 1024 = 2^{10}$$

s.t.

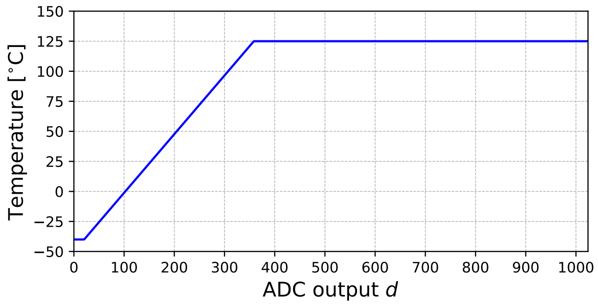
$$d = \left\lfloor \frac{V_{in} \cdot 2^{10}}{V_{ref}} \right\rfloor = \left\lfloor \frac{1024\,V_{in}}{5\,\mathrm{V}} \right\rfloor$$

Models of Sensors

To obtain T, map return value d of `analogRead` to T by inverting (1) the ADC quantization with $V_{in} = v$ and (2) the sensor model:

$$v \approx 5d/1024$$

$$T \approx \begin{cases} -40\,, & v < 0.1 \\ 100v - 50\,, & 0.1 \leq v \leq 1.75 \\ 125\,, & v > 1.75 \end{cases}$$



Models of Sensors

Arduino sketch: Update with TMP36 model

```
float Tmap(int d) {
  float v = 5. * d / 1024.;
  if (v < 0.1)
    return -40.;
  else if (v <= 1.75)
    return 100. * v - 50.;
  else
    return 125.;                // T_u = 125°C
}

void loop() {
  int temp = analogRead(A0);    // sample ADC output
  Serial.print("Temp: ");       // send value to host
  Serial.println(Tmap(temp));
  delay(1000);                  // wait 1s
}
```


Dynamic Range

The dynamic range of an analog sensor with operating range

$$x_l \leq x \leq x_u$$

is defined in decibels as

$$D = 20 \log_{10} \frac{x_u}{x_l}$$

Example: TMP36 operating range $T_l = -40\,°C < T < T_u = 150\,°C$.

Since $\log x$ is defined for $x > 0$, convert to a unit with nonnegative temperature values, e.g. Kelvin:

$$D = 20 \log_{10} \frac{T_u}{T_l} = 20 \log_{10} \frac{423.15\,K}{233.15\,K} = 5.18\,dB$$

Dynamic Range

Quantization due to ADC incurs a loss of accuracy: if two values of the physical quantity are too close to each other, they become indistinguishable.

The precision p of a sensor, aka resolution in the context of ADCs or timers, is the smallest spacing between two values of the physical quantity whose digital sensor readings (after ADC) are distinguishable.

The dynamic range of a digital sensor with operating range $x_l \leq x \leq x_u$ is defined in decibels as

$$D = 20 \log_{10} \frac{x_u - x_l}{p}$$

Dynamic Range

Example: 10-bit ADC with $V_{ref} = 5\,V$ has precision

$$p_{ADC} = 2^{-10} V_{ref} = \frac{5\,V}{1024} = 4.88\,mV$$

Within the operating range of the ADC, sensor voltage v and ADC output d are related by

$$v = p_{ADC}\, d$$

The precision of the TMP36 sensor is the smallest spacing between temperature values associated with discrete values d and $d+1$.

Apply TMP36 model to find the precision in terms of temperature:

$$\begin{aligned} p_T &= T_{d+1} - T_d = (100 v_{d+1} - 50) - (100 v_d - 50) \\ &= 100 p_{ADC} = 0.488°C \end{aligned}$$

Dynamic Range

Example cont'd: Dynamic range of digital temperature sensor, consisting of analog TMP36 and ADC:

$$D \; = \; 20 \log_{10} \frac{T_u - T_l}{p_T} \; = \; 20 \log_{10} \frac{125 - (-40)}{0.488} \; = \; 50.6\,\text{dB}$$

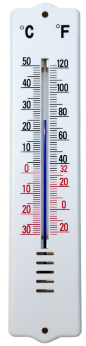
Note:

- The smaller the precision the more accurate is a digital sensor.
- The dynamic range reflects accuracy: The more accurate a digital sensor is, the smaller is its precision, and the larger is its dynamic range.

Calibration

Calibration of a sensor compares actual sensor readings with those of a calibration standard of known accuracy.

Example: Compare the temperature sample of the TMP36 sensor with the reading of a thermometer.



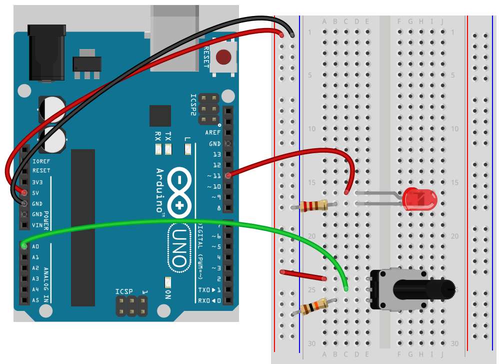
Calibration facilitates

- comparison and reproducibility of independent measurements,
- adjustments to correct errors,
- establishing the operating range of a sensor.

Note: The TMP36 sensor is (nominally) accurate to within 0.1°C without calibrating adjustments.

Calibration

Task: Calibrate the potentiometer so as to control the brightness of the LED across its full range.



analogWrite: PWM parameter range is $[0, 255]$
analogRead: 10-bit ADC range is $[0, 1023]$
A0 voltage: voltage divider: 10k pot and 10k resistor

Calibration

Circuit analysis: extreme positions of potentiometer

R_{pot} [kΩ]	V_{A0} [V]	d
0	5	1023
10	2.5	511

Calibration: Measure $d_{\min}$ and $d_{\max}$ in face of imperfect components.

Task: Map operating range $[d_{\min}, d_{\max}]$ to PWM domain $[0, 255]$ for adjusting the LED brightness.

Calibration

Arduino sketch: measure operating range of pot

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int pot = analogRead(A0);

  Serial.print("Pot: ");      // send value to host
  Serial.println(pot);

  delay(1000);                // wait 1s
}
```

Instructions: Turn pot to farmost left and right positions, and record d_{min} and d_{max}.

Calibration

Map operating range: pot sample (analogRead) to PWM domain

```
int Pmap(int d) {
  int m = (int) (((d - potMin) * 255.) / (potMax - potMin));

  if (m < 0)
    return 0;
  else if (m <= 255)
    return m;
  else
    return 255;
}
```

Presumption: d_{min} is stored in global variable potMin and d_{max} in global variable potMax.

Defensive programming: Pmap always returns a value inside PWM domain $[0, 255]$.

Calibration

Semiautomated calibration: calibrate in setup by turning pot

```
int potMin = 1023;                // = d_min
int potMax = 0;                   // = d_max

void calibratePot() {
  int n;
  for (n = 0; n < 1000; n++) {    // take 1000 samples
    int pot = analogRead(A0);     // sample ADC output

    if (pot < potMin)
      potMin = pot;               // update = d_min
    if (pot > potMax)
      potMax = pot;               // update = d_max

    delay(10);        // 10 seconds total for calibration
  }
}
```


Calibration

Arduino sketch: control LED brightness with pot

```
void setup() {
  Serial.begin(9600);
  calibratePot();              // calibrate upon startup
}

void loop() {
  int pot = analogRead(A0);    // sample pot
  int led = Pmap(pot);         // map pot sample to PWM domain

  Serial.print("Pot: ");       // send values to host
  Serial.print(pot);
  Serial.print(" ");
  Serial.println(led);

  analogWrite(11, led);        // set LED brightness

  delay(1000);                 // wait 1s
}
```