HTL | MÖSSINGERSTRASSE

Digitale Regler

Version 1.2

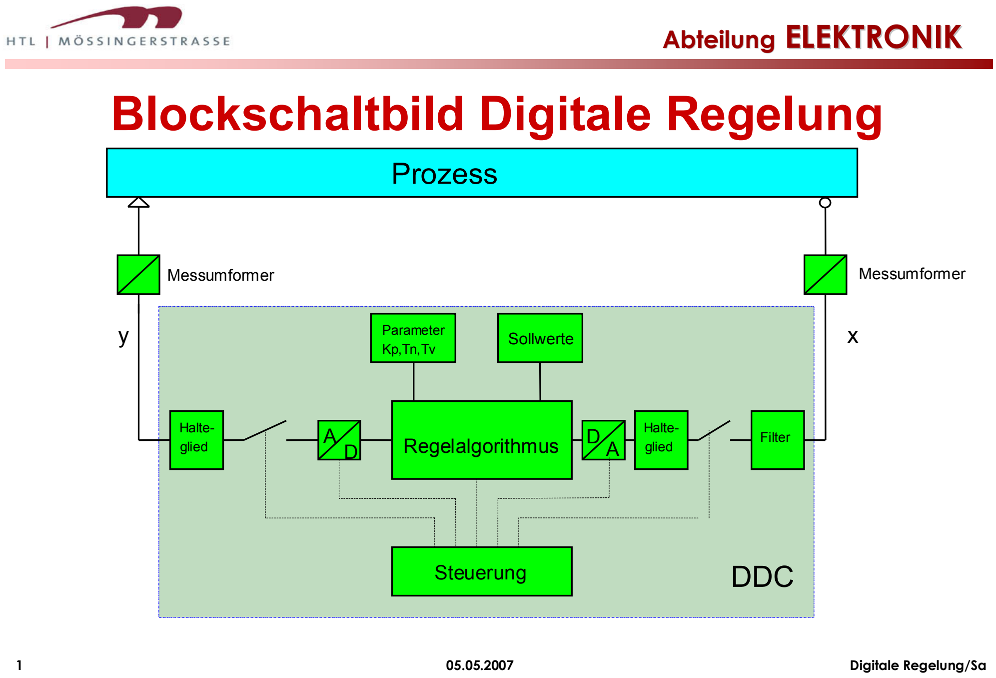
HTL Mössingerstrasse, Abt. Elektronik
September 2008

Inhaltsverzeichnis

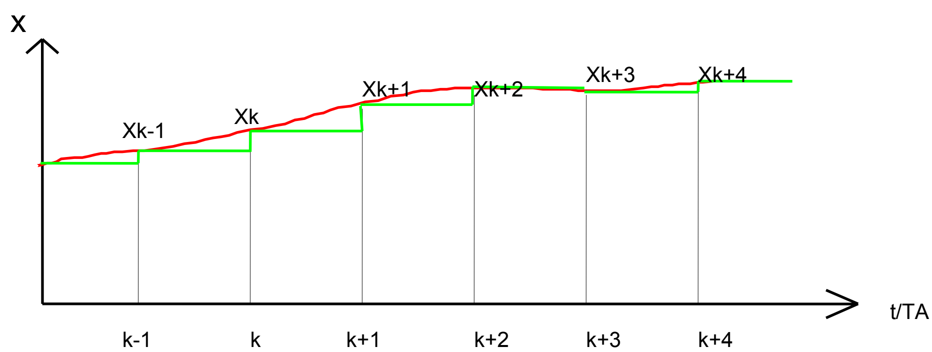
1 Digitale Regler

1.1 Grundlagen

Digitale Regelung, DDC-Direct Digital Control



In der analogen Regelungstechnik wird die Reglerfunktion durch Beschalten eines Operationsverstärkers mit Widerständen und Kondensatoren erzeugt.
Bei der digitalen Regelung berechnet ein Programm nach einem Regelalgorithmus die benötigte Stellgrößenfolge y_k.
Das kontinuierliche Eingangssignal wird abgetastet (Abtast-Halteglied) und zur Weiterverarbeitung wird die dabei entstehende Treppenfunktion verwendet.



Der A/D Wandler liefert dann einen Zahlenwert x_k der mit dem eingegebenen Sollwert w_k verglichen wird.

Für die Regeldifferenz e_k gilt: $\mathbf{e_k = w_k - x_k}$

1.2 Basisalgorithmen für digitale Regelungen

1.2.1 Proportionalalgorithmus

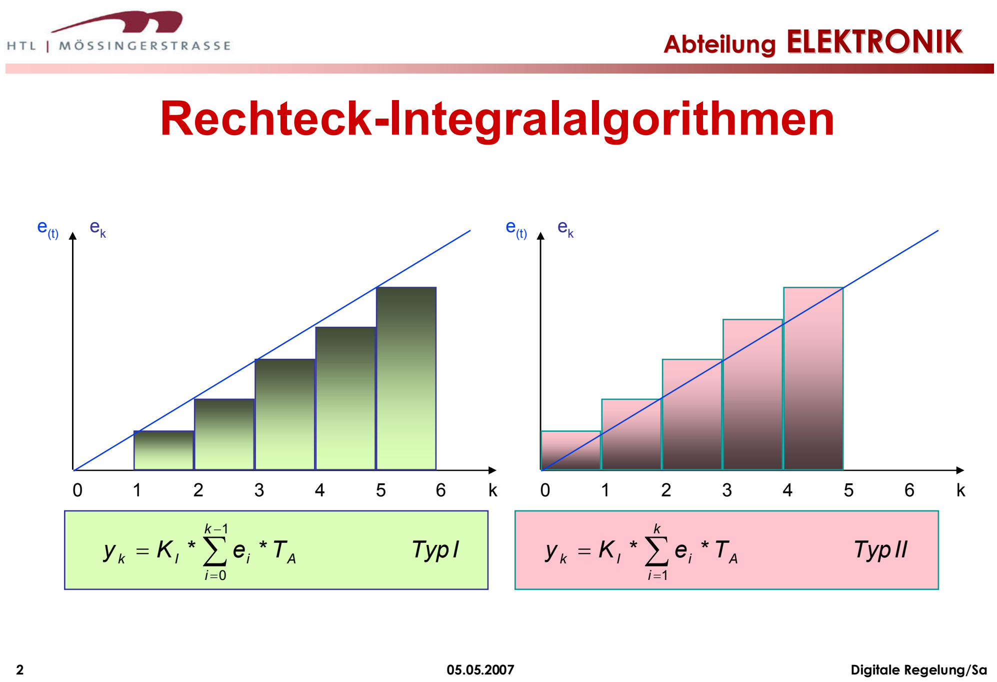
Aus den aktuellen Abtastwerten x_k und w_k wird der Error e_k berechnet und mit der Proportionalverstärkung K_P multipliziert.
Das Ergebnis ergibt dann die aktuelle Stellgröße y_k.

$$e_k = w_k - x_k$$
$$y_k = K_P * e_k$$
$$k = k * T_A$$

1.2.2 Integralalgorithmen

Die kontinuierliche Reglerfunktion Integration kann auf diskrete Werte e_k nicht angewendet werden und muss durch die diskreten Funktionen Addition und Subtraktion angenähert werden.

- **Rechtecknäherung:**

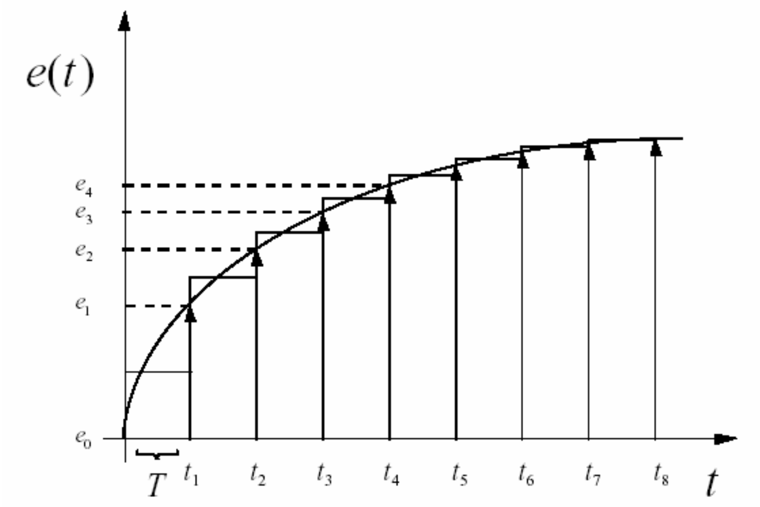


$$y_k = K_I * \sum_{i=0}^{k-1} e_i * T_A \qquad Typ\,I$$

$$y_k = K_I * \sum_{i=1}^{k} e_i * T_A \qquad Typ\,II$$

Typ I liefert zu kleine Werte und Typ II liefert zu große Werte. Wenn die Abtastzeit T_A hinreichend klein ist, wird der Unterschied vernachlässigbar.

- **Trapeznäherung:**



$$y_k = K_I * \sum_{i=1}^{k} \frac{1}{2}[e_i + e_{i-1}] * T_A = K_I * T_A * \frac{1}{2} * \sum_{i=1}^{k} [e_i + e_{i-1}]$$

Um die Rechenzeit zu verkürzen und den Speicherbedarf zu verringern wird ein rekursiver Algorithmus zur Berechnung der Stellgröße eingeführt.

- **Rekursiver Algorithmus Typ I**

$$y_k = K_I * \sum_{i=0}^{k-1} e_i * T_A \qquad y_{k-1} = K_I * \sum_{i=0}^{k-2} e_i * T_A \qquad TypI$$

$$y_k - y_{k-1} = K_I * \sum_{i=0}^{k-1} e_i * T_A - K_I * \sum_{i=0}^{k-2} e_i * T_A = K_I * T_A * e_{k-1}$$

$$y_k = y_{k-1} + K_I * T_A * e_{k-1} = y_{k-1} + \frac{T_A}{T_I} * e_{k-1}$$

- **Rekursiver Algorithmus Typ II**

$$y_k = K_I * \sum_{i=1}^{k} e_i * T_A \qquad y_{k-1} = K_I * \sum_{i=1}^{k-1} e_i * T_A \qquad TypII$$

$$y_k - y_{k-1} = K_I * \sum_{i=1}^{k} e_i * T_A - K_I * \sum_{i=1}^{k-1} e_i * T_A = K_I * T_A * e_k$$

$$y_k = y_{k-1} + K_I * T_A * e_k = y_{k-1} + \frac{T_A}{T_I} * e_k$$

- **Rekursiver Algorithmus Trapeznäherung**

$$y_k = K_I * \sum_{i=1}^{k} \frac{1}{2}\left[e_i + e_{i-1}\right] * T_A = K_I * T_A * \frac{1}{2} * \sum_{i=1}^{k}\left[e_i + e_{i-1}\right]$$

$$y_{k-1} = K_I * T_A * \frac{1}{2} * \sum_{i=1}^{k-1}\left[e_i + e_{i-1}\right]$$

$$y_k - y_{k-1} = K_I * T_A * \frac{1}{2} * \sum_{i=1}^{k}\left[e_i + e_{i-1}\right] - K_I * T_A * \frac{1}{2} * \sum_{i=1}^{k-1}\left[e_i + e_{i-1}\right]$$

$$y_k - y_{k-1} = K_I * T_A * \frac{1}{2} * \left(e_k + e_{k-1}\right)$$

$$y_k = y_{k-1} + \frac{T_A}{2T_I} * \left(e_k + e_{k-1}\right)$$

Nur der jeweils letzte Wert der Stellgröße und die beiden letzten Werte von e müssen gespeichert werden.
Die Abtastzeit T_A geht in die Berechnung von y_k ein. Daher ist bei einer Implementierung darauf zu achten, dass das Abtastintervall T_A genau eingehalten wird.
(Realisierung mit Echtzeituhr und Interrupt.)

1.2.3 Differentialalgorithmen

Die kontinuierliche Operation Differenzieren wird bei der digitalen Regelung durch eine Differenzenbildung angenähert. Aus einem Differentialquotienten entsteht damit ein Differenzenquotient.

$$y_{(t)} = K_D \frac{de_{(t)}}{dt} \rightarrow y_k = \frac{\Delta e_k}{\Delta t}$$

Als Zeitdifferenz Δt wird die Abtastperiode T_A gewählt.

- **Differenzenbildung mit der linken Intervallgrenze Typ I (rückwärts)**

$$y_k = \frac{K_D}{T_A} * \left(e_k - e_{k-1}\right)$$

- **Differenzenbildung mit der rechten Intervallgrenze Typ II (vorwärts)**

$$y_k = \frac{K_D}{T_A} * \left(e_{k+1} - e_k\right)$$

Da der Wert e_{k+1} zum Abtastzeitpunkt k noch nicht vorliegt, kann dieser Algorithmus nur das y_{k-1} zum Zeitpunkt k berechnen.

Der Algorithmus nach Typ I liefert zu kleine Werte und der Typ II zu große Werte.

Bei digitalen Reglern wird meist der Typ I verwendet.
Bei stark gestörten Messgrößen wird auch der Mittelwert-Algorithmus angewandt.

- **Differentialalgorithmus mit Mittelwertbildung**

$$y_k = \left[\frac{K_D}{T_A} * \left(e_{k+1} - e_k\right) + \frac{K_D}{T_A} * \left(e_k - e_{k-1}\right)\right] * \frac{1}{2} = \frac{K_D}{2T_A} * (e_{k+1} - e_{k-1})$$

$$y_{k-1} = \frac{K_D}{2T_A} * (e_k - e_{k-2})$$

Bsp.: Differentiation 2. Ordnung

$$\frac{d\overset{2}{e}_{(t)}}{dt} \cong \frac{\frac{e_k - e_{k-1}}{T_A} - \frac{e_{k-1} - e_{k-2}}{T_A}}{T_A} = \frac{e_k - 2e_{k-1} + e_{k-2}}{T_A}$$

Bsp.: Verzögerung 1. Ordnung

Eingang: x

Ausgang: y

$$\frac{x}{y} = \frac{1}{1 + s * T_1}$$

$$x + s * T_1 * x = y$$

$$x_k + T_1 * \frac{x_k - x_{k-1}}{T_A} = y_k$$

$$x_k * T_A + T_1 * x_k - T_1 * x_{k-1} = y_k * T_A$$

$$x_k * \left(T_A + T_1\right) - T_1 * x_{k-1} = y_k * T_A$$

$$x_k = \frac{y_k * T_A + T_1 * x_{k-1}}{T_A + T_1}$$

$$x_k = \frac{T_A}{T_A + T_1} * y_k + \frac{T_1}{T_A + T_1} * x_{k-1}$$

1.3 Regelalgorithmen für Standardregler

1.3.1 PID-Stellungsalgorithmus

Die analoge PID-Reglerfunktion in Parallelform lautet wie folgt:

$$y_{(t)} = K_P * \left(e_{(t)} + \frac{1}{Tn} \int_0^t e_{(t)} dt + T_v \frac{de_{(t)}}{dt} \right)$$

Die diskrete nichtrekursive Form des PID-Regelalgorithmus Typ I (Stellungsalgorithmus) lautet:

$$y_k = K_P * \left(e_k + \frac{T_A}{T_n} \sum_{i=0}^{k-1} e_i + \frac{T_v}{T_A} \left(e_k - e_{(k-1)} \right) \right)$$

$$k = \frac{t}{T_A} \ldots diskrete\ Zeiteinheit\ 0{,}1{,}2{,}3\ldots$$

$$T_A \ \ldots Abtastzeit$$

Wenn man den Stellungsalgorithmus in vorheriger Form direkt programmiert, müsste man zur Berechnung immer alle vergangenen Regelabweichungen benötigen.
Das würde aber mit jedem Abtastschritt zu einem Anwachsen der Speichertiefe und zu einer Verlängerung der Rechenzeit führen.
Es ist daher vernünftiger auf rekursive Algorithmen zurückzugreifen, da dadurch der Speicherplatzbedarf reduziert und die Berechnungsgeschwindigkeit erhöht wird.

Um die rekursive Formel für den Stellungsalgorithmus zu erhalten, stellt man die Gleichung für y_{k-1} auf.

$$y_{(k-1)} = K_P * \left(e_{(k-1)} + \frac{T_A}{T_n} \sum_{i=0}^{k-2} e_i + \frac{T_v}{T_A} \left(e_{(k-1)} - e_{(k-2)} \right) \right)$$

Subtraktion

$$y_k - y_{(k-1)} = K_P * \left(e_k - e_{(k-1)} + \frac{T_A}{T_n} \left(\sum_{i=0}^{k-1} e_i - \sum_{i=0}^{k-2} e_i \right) + \frac{T_v}{T_A} \left(e_k - e_{(k-1)} - e_{(k-1)} + e_{(k-2)} \right) \right)$$

$$y_k = y_{(k-1)} + e_k * \left(K_P + K_P \frac{T_V}{T_A} \right) - e_{(k-1)} * \left(K_P + 2K_P \frac{T_v}{T} - K_P \frac{T_A}{T_n} \right) + e_{(k-2)} * \left(K_P \frac{T_V}{T_A} \right)$$

$$y_k = y_{(k-1)} + q_0 e_k + q_1 e_{(k-1)} + q_2 e_{(k-2)}$$

$$q_0 = K_P * \left(1 + \frac{T_V}{T_A} \right)$$

$$q_1 = -K_P * \left(1 + 2\frac{T_v}{T_A} - \frac{T_A}{T_n} \right)$$

$$q_2 = K_P * \frac{T_V}{T_A}$$

1.3.2 PID-Geschwindigkeitsalgorithmus

$$\Delta y_k = y_k - y_{(k-1)}$$

$$\Delta y_k = +q_0 e_k + q_1 e_{(k-1)} + q_2 e_{(k-2)}$$

$$q_0 = K_P * \left(1 + \frac{T_V}{T_A}\right)$$

$$q_1 = -K_P * \left(1 + 2\frac{T_v}{T_A} - \frac{T_A}{T_n}\right)$$

$$q_2 = K_P * \frac{T_V}{T_A}$$

Der Geschwindigkeitsalgorithmus berechnet nur die Änderung der Stellgröße für jeden Abtastzeitpunkt. Die Änderung Δy_k bezogen auf die Abtastzeit T_A, entspricht einer Stellgeschwindigkeit.

Unterschied Stellungs- und Geschwindigkeitsalgorithmus
Beim Stellungsalgorithmus wird im Regelalgorithmus integriert und beim Geschwindigkeitsalgorithmus muss das Stellglied integrales Verhalten haben (Schrittmotor).

1.3.3 Modifizierte PID-Algorithmen

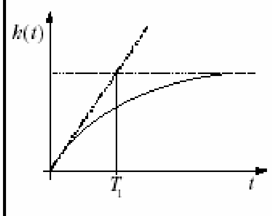
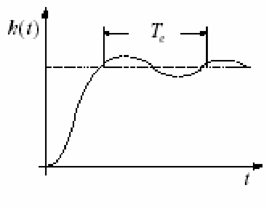
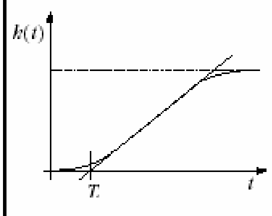
Bei Sollwertsprüngen entstehen durch den D-Anteil große Stellgrößen welche die Stellglieder belasten und den Regelkreis zum Schwingen anregen können.
Abhilfe wenn man beim D-Anteil e =w-x durch -x ersetzt.

$$y_k - y_{(k-1)} = K_P * \left(e_k - e_{(k-1)} + \frac{T_A}{T_n}\left(\sum_{i=0}^{k-1} e_i - \sum_{i=0}^{k-2} e_i\right) + \frac{T_v}{T_A}\left(e_k - e_{(k-1)} - e_{(k-1)} + e_{(k-2)}\right)\right)$$

$$y_k = y_{(k-1)} + K_P * \left[e_k - e_{k-1} + \frac{T_A}{T_n} * e_{k-1} - \frac{T_v}{T_A}\left(x_k - 2x_{k-1} + x_{k-2}\right)\right]$$

1.3.4 Auswahl der Abtastzeit Ta

Ermittlung von Ta um quasikontinuierlich arbeitende digitale Regler zu erhalten.
Sprungantwort der Strecke:

Sprungantwort			
Wahl der Abtastzeit	$T \leq 0{,}1 \cdot T_1$	$T \leq 0{,}1 \cdot T_e$	$T \leq 0{,}25 \cdot T_t$

1.4 Implementierungsbeispiel eines PID - Reglers

Istwert einlesen über ADC0 , Stellgröße über PWM ausgeben.

```
#include <18f458.h>
#device ADC=10
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=pin_C6, rcv=pin_C7, Parity=N, BITS=8)
// ---------- Deklaration der Variablen ----------
int16 istwert = 0;
int16 sollwert = 0;
float sollwertx;
float yk = 0;
float yk1 = 0;
signed int16 duty;
float kp = 1.3;
float Tn = 230;
float Ta = 50;
float Tv = 80;
float q0 = 0;
float q1 = 0;
float q2 = 0;
float error = 0;
float error1 = 0;
float error2 = 0;
char eingabe=0;

// ---------- Deklarieren der Unterprogramme ----------
void ccp1init();
void adcinit();
void interruptsenable();
```

```
// ----------  ISR Timer0 ----------
#INT_TIMER0
isr_timer0()
{
        set_adc_channel(0);
        delay_us(10);
        istwert = read_adc();

        yk1=yk;
        error2=error1;
        error1=error;
        error = ((float)istwert - sollwert);
        yk = yk1 + q0*error + q1*error1 + q2*error2;
        duty = (signed int16)yk;

        // Stellgröße auf maximal 515 / minimal 0 begrenzen
        if(duty >= 515) {duty = 515;}
        if(duty <= 0) {duty=0;}

        // PWM - Ausgabe
        set_pwm1_duty(duty);
        set_timer0(50000);
}
```

```
main()
{
        // Aufrufen der Unterprogramme
        ccp1init();
        adcinit();
        interruptsenable();

        // Definieren der Konstanten
        q0 = kp + ((kp*tv)/ta);
        q1 = (kp*ta)/tn - (2*kp*tv)/ta - kp;
        q2 = (kp*tv)/ta;

        while(true)
        {
                if(kbhit())
                {
                        disable_interrupts(INT_TIMER0);
                        eingabe = getc();
                        sollwertx = ((float)eingabe*5.115);
                        sollwert = ((long)sollwertx);
                        enable_interrupts(INT_TIMER0);
                }
        }

}



// CCP1 einstellen
void ccp1init()
{
        setup_ccp1(CCP_PWM);
        setup_timer_2(T2_DIV_BY_16, 127, 1);
}
```

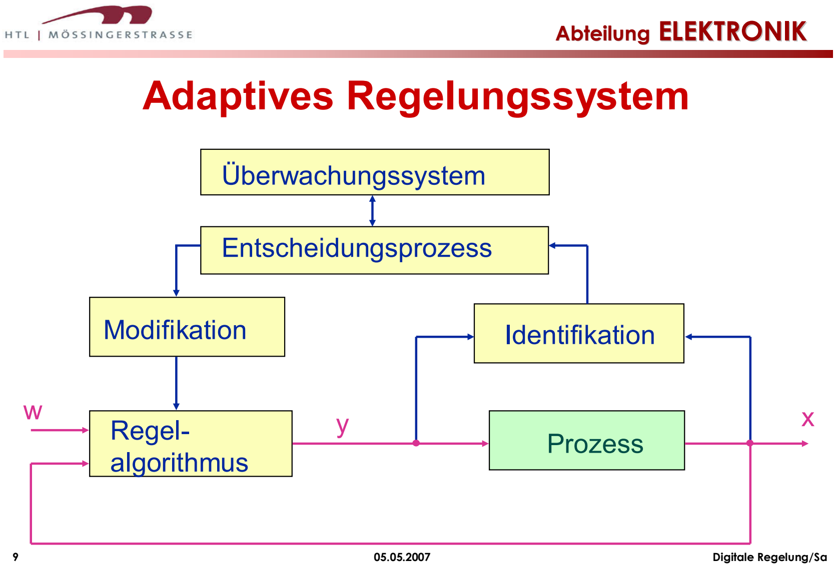
```
// ADC - einstellen
void adcinit()
{
        setup_adc_ports(RA0_ANALOG);
        setup_adc(adc_clock_internal);
}


// Interrupts aktivieren
void interruptsenable()
{
        enable_interrupts(GLOBAL);
        setup_timer_0(RTCC_INTERNAL | RTCC_DIV_8);
        set_timer0(50000);
        enable_interrupts(INT_TIMER0);

}
```

1.5 Selbsteinstellende-, Adaptive-Regelungsverfahren

Durch die Einführung von digitalen Reglern (Prozess- oder Kompaktregler) wurde es softwaretechnisch möglich noch zusätzliche Funktionen im Regler zu implementieren.
Eine der wichtigsten Zusatzfunktionen ist die automatische Anpassung der Reglerparameter an das Streckenverhalten.



Gliederung in drei Ebenen: Überwachungsebene, Adaptionsebene, Regelungsebene

- **Identifikation**

Ermittlung der Streckenparameter aus gemessenen Ein- und Ausgangsgrößen.

- **Entscheidungsprozess**

Nach Gütekriterien wird entschieden wie auf die Identifikationsergebnisse mit Hilfe einer Modifikation reagiert wird.

Integrale Gütekriter ien

$$ISE : G = \int_0^\infty e_{(t)}^2 * dt$$

$$ITAE : G = \int_0^\infty e_{(t)} * t * dt$$

$$Allg. : G = \int_0^\infty \left(e_{(t)}^2 + r\Delta y_{(t)}\right) * dt$$

- **Modifikation**

Beeinflussung der Reglerfunktion bezüglich Struktur (P,PI,PID) und Parametern (K,Tn,Tv).

- **Überwachung**

Aus Gründen der Zuverlässigkeit ist hier die Funktionsüberwachung und die Koordination der Teilsysteme bzw. Gesamtsystems realisiert. Bei Fehlererkennung werden entsprechende Schritte gesetzt.