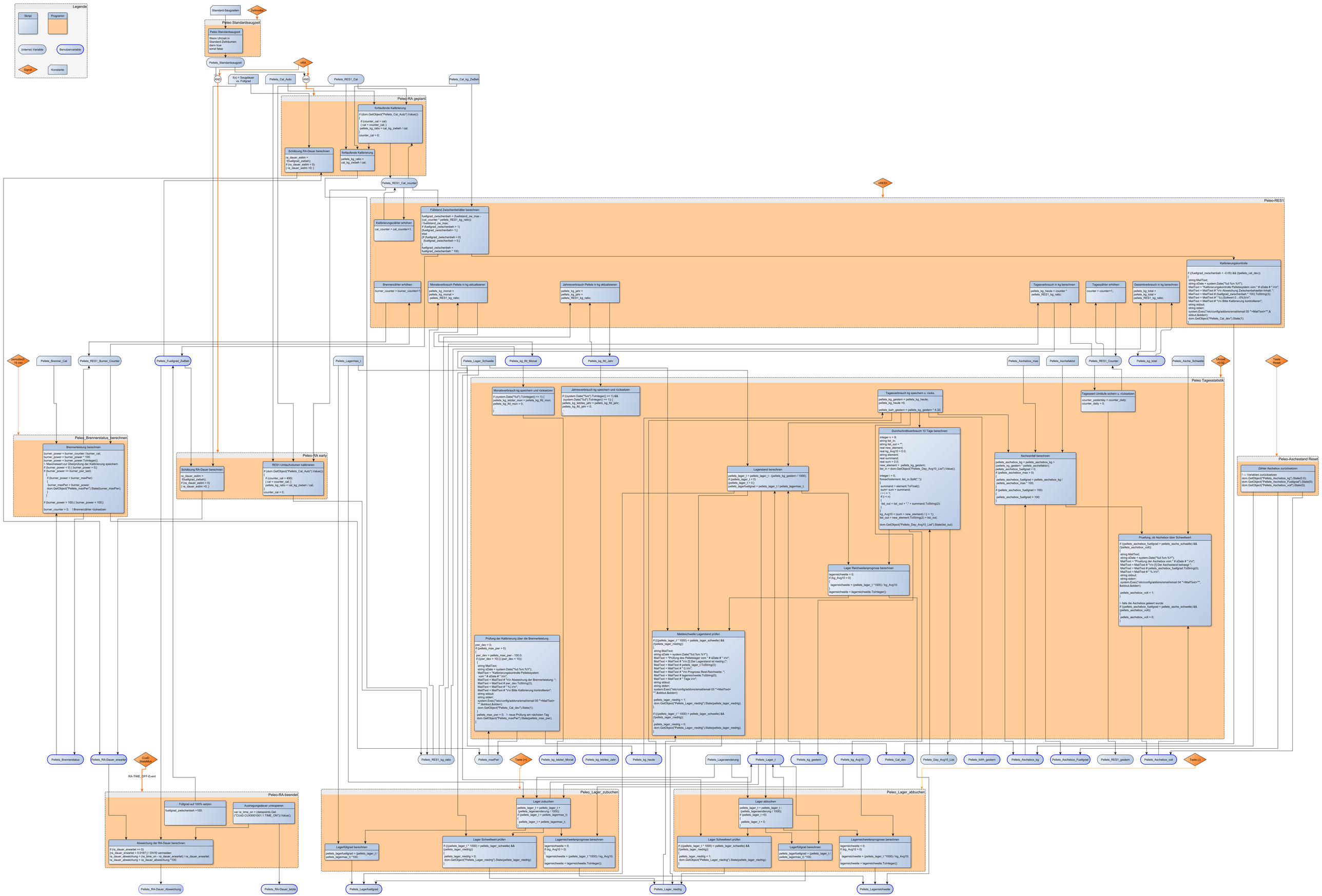
Legende
Skript
Programm
(interne) Variable
Benutzervariable
Signal
Konstante
Standard-Saugzeiten
Zeitmodul
Peleo Standardsaugzeit
Peleo Standardsaugzeit
Wenn Uhrzeit in Standard-Zeitrlumen dann true sonst false
Pellets_Standardsaugzeit
rRA
AND
f(x) = Saugdauer vs. Füllgrad
Pellets_Cal_Auto
AND
Pellets_RES1_Cal
Pellets_Cal_kg_Zwßeh
Peleo-RA geplant
fortlaufende Kalibrierung
if (dom.GetObject("Pellets_Cal_Auto").Value())
{
if (counter_cal > cal)
{ cal = counter_cal; }
pellets_kg_ratio = cal /kg_zwbeh / cal;
}
counter_cal = 0;
Schätzung RA-Dauer berechnen
ra_dauer_estim =
f(fuellgrad_zwbeh);
if (ra_dauer_estim < 0)
{ ra_dauer_estim =0; }
fortlaufende Kalibrierung
pellets_kg_ratio =
cal_kg_zwbeh / cal;
Pellets_RES1_Cal_counter
rRES1
Peleo-RES1
Füllstand Zwischenbehälter berechnen
fuellgrad_zwischenbeh = (fuellstand_zw_max -
(cal_counter * pellets_RES1_kg_ratio))
/ fuellstand_zw_max;
if (fuellgrad_zwischenbeh > 1)
{fuellgrad_zwischenbeh= 1;}
else
{if (fuellgrad_zwischenbeh < 0)
{fuellgrad_zwischenbeh = 0;}
}
fuellgrad_zwischenbeh =
fuellgrad_zwischenbeh * 100;
Kalibrierungszähler erhöhen
cal_counter = cal_counter+1;
Kalibrierungskontrolle
if ((fuellgrad_zwischenbeh < -0.05) && !(pellets_cal_dev))
{
string MailText;
string sDate = system.Date("%d.%m.%Y");
MailText = "Kalibrierungskontrolle Pelletssystem vom " # sDate # ":\n";
MailText = MailText # "\n Abweichung Zwischenbehälter-Inhalt: ";
MailText = MailText # (fuellgrad_zwischenbeh * 100).ToString(3);
MailText = MailText # " %.\n Sollwert 0 ...-5%.\n\n";
MailText = MailText # "\n\n Bitte Kalibrierung kontrollieren";
string stdout;
string stderr;
system.Exec("/etc/config/addons/email/email 05 "+MailText+"" &
stdout,&stderr);
dom.GetObject("Pellets_Cal_dev").State(1);
}
Brennerzähler erhöhen
burner_counter = burner_counter+1;
Monatsverbrauch Pellets in kg aktualisieren
pellets_kg_monat =
pellets_kg_monat +
pellets_RES1_kg_ratio;
Jahresverbrauch Pellets in kg aktualisieren
pellets_kg_jahr =
pellets_kg_jahr +
pellets_RES1_kg_ratio;
Tagesverbrauch in kg berechnen
pellets_kg_heute = counter *
pellets_RES1_kg_ratio;
Tageszähler erhöhen
counter = counter+1;
Gesamtverbrauch in kg berechnen
pellets_kg_total =
pellets_kg_total +
pellets_RES1_kg_ratio;
Pellets_Brenner_Cal
Pellets_RES1_Burner_Counter
Pellets_Fuellgrad_Zwßeh
Pellets_Lagermax_t
Pellets_Lager_Schwelle
Pellets_kg_ltzt_Monat
Pellets_kg_ltzt_Jahr
Pellets_Aschebox_max
Pellets_Aschefaktor
Pellets_RES1_Counter
Pellets_kg_total
Pellets_Asche_Schwelle
Peleo Tagesstatistik
Monatsverbrauch kg speichern und rücksetzen
if (system.Date("%d",ToInteger() == 1) {
pellets_kg_letzter_mon = pellets_kg_ltzt_mon;
pellets_kg_ltzt_mon = 0;
...
}
Jahresverbrauch kg speichern und rücksetzen
if ((system.Date("%m"),ToInteger() == 1) &&
(system.Date("%d",ToInteger() == 1)) {
pellets_kg_letztes_jahr = pellets_kg_ltzt_jahr;
pellets_kg_ltzt_jahr = 0;
...
}
Tagesverbrauch kg speichern u. rücks.
pellets_kg_gestern = pellets_kg_heute;
pellets_kg_heute =0;
pellets_kwh_gestern = pellets_kg_gestern * 4.32;
Tageswert,Umläufe sichern u. rücksetzen
counter_yesterday = counter_daily;
counter_daily = 0;
Durchschnittsverbrauch 10 Tage berechnen
integer n = 9;
string list_in;
string list_out = "";
real new_element;
real kg_Avg10 = 0.0;
string element;
real summand;
real sum = 0.0;
new_element = pellets_kg_gestern;
list_in = dom.GetObject("Pellets_Day_Avg10_List").Value();
integer i = 0;
foreach(element, list_in.Split(";"))
{
summand = element.ToReal();
sum= sum + summand;
i= i + 1;
if (i < n)
{
list_out = list_out + "," + summand.ToString(2);
}
}
kg_Avg10 = (sum + new_element) / (i + 1);
list_out = new_element.ToString(2) + list_out;
dom.GetObject("Pellets_Day_Avg10_List").State(list_out)
Lagerstand berechnen
pellets_lager_t = pellets_lager_t - (pellets_kg_gestern / 1000);
if (pellets_lager_t < 0)
{pellets_lager_t= 0;}
pellets_lagerfuellgrad = pellets_lager_t / pellets_lagermax_t;
Ascheanfall berechnen
pellets_aschebox_kg = pellets_aschebox_kg +
(pellets_kg_gestern * pellets_aschefaktor);
pellets_aschebox_fuellgrad = 0;
if (pellets_aschebox_max > 0)
{
pellets_aschebox_fuellgrad = pellets_aschebox_kg /
pellets_aschebox_max * 100;
}
if (pellets_aschebox_fuellgrad > 100)
{
pellets_aschebox_fuellgrad = 100;
}
Prüfung, ob Aschebox über Schwellwert
if ((pellets_aschebox_fuellgrad > pellets_asche_schwelle) &&
(pellets_aschebox_voll))
{
string MailText;
string sDate = system.Date("%d.%m.%Y");
MailText = "Prüefung der Aschebox vom " # sDate # ":\n";
MailText = MailText # "\n [!] Der Aschestand betraegt ";
MailText = MailText # pellets_aschebox_fuellgrad.ToString(0);
MailText = MailText # " %.\n";
string stdout;
string stderr;
system.Exec("/etc/config/addons/email/email 04 "+MailText+"",
&stdout,&stderr);
pellets_aschebox_voll = 1;
}
\- falls die Aschebox geleert wurde
if ((pellets_aschebox_fuellgrad < pellets_asche_schwelle) &&
(pellets_aschebox_voll))
{
pellets_aschebox_voll = 0;
}
Lager Reichweitenprognose berechnen
lagerreichweite = 0;
if (kg_Avg10 > 0)
{
lagerreichweite = (pellets_lager_t *1000) / kg_Avg10;
}
lagerreichweite = lagerreichweite.ToInteger();
Prüfung der Kalibrierung über die Brennerleistung
pwr_dev = 0;
if (pellets_max_pwr > 0)
{
pwr_dev = pellets_max_pwr - 100.0;
if ((pwr_dev > 10) || (pwr_dev < 10))
{
string MailText;
string sDate = system.Date("%d.%m.%Y");
MailText = "Kalibrierungskontrolle Pelletssystem
vom " # sDate # ":\n\";
MailText = MailText # "\n Abweichung der Brennerleistung: ";
MailText = MailText # pwr_dev.ToString(3);
MailText = MailText # " %.\n\";
MailText = MailText # "\n Bitte Kalibrierung kontrollieren";
string stdout;
string stderr;
system.Exec("/etc/config/addons/email/email 05 "+MailText+
"",&stdout,&stderr);
dom.GetObject("Pellets_Cal_dev").State(1);
}
pellets_max_pwr = 0; \- neue Prüfung am nächsten Tag
dom.GetObject("Pellets_maxPwr").State(pellets_max_pwr);
}
Meldeschwelle Lagerstand prüfen
if (((pellets_lager_t * 1000) < pellets_lager_schwelle) &&
(!pellets_lager_niedrig))
{
string MailText;
string sDate = system.Date("%d.%m.%Y");
MailText = "Prüfung des Pelletslager vom " # sDate # ":\n\";
MailText = MailText # "\n [!] Der Lagerstand ist niedrig (";
MailText = MailText # pellets_lager_t.ToString(3);
MailText = MailText # " t).\n\";
MailText = MailText # "\n\n Prognose Rest-Reichweite: ";
MailText = MailText # lagerreichweite.ToString(0);
MailText = MailText # " Tage.\n\";
string stdout;
string stderr;
system.Exec("/etc/config/addons/email/email 03 "+MailText+
"",&stdout,&stderr);
pellets_lager_niedrig = 1;
dom.GetObject("Pellets_Lager_niedrig").State(pellets_lager_niedrig);
}
if (((pellets_lager_t * 1000) > pellets_lager_schwelle) &&
(!pellets_lager_niedrig))
{
pellets_lager_niedrig = 0;
dom.GetObject("Pellets_Lager_niedrig").State(pellets_lager_niedrig);
}
Peleo-Aschestand Reset
Zähler Aschebox zurücksetzen
! --- Variablen zurücksetzen
dom.GetObject("Pellets_Aschebox_kg").State(0.0);
dom.GetObject("Pellets_Aschebox_Fuellgrad").State(0);
dom.GetObject("Pellets_Aschebox_voll").State(0);
Peleo_Brennerstatus_berechnen
Brennerleistung berechnen
burner_power = burner_counter / burner_cal;
burner_power = burner_power * 100;
burner_power = burner_power.ToInteger();
\- Maximalwert zur Überprüfung der Kalibrierung speichern
if (burner_power < 0) { burner_power = 0;}
if (burner_power >= burner_pwr_last)
{
if (burner_power > burner_maxPwr)
{
burner_maxPwr = burner_power;
dom.GetObject("Pellets_maxPwr").State(burner_maxPwr);
}
}
if (burner_power > 100) { burner_power = 100;}
burner_counter = 0; ! Brennerzähler rücksetzen
Peleo-RA early
Schätzung RA-Dauer berechnen
ra_dauer_estim =
f(fuellgrad_zwbeh);
if (ra_dauer_estim < 0)
{ ra_dauer_estim =0; }
RES1-Umlaufvolumen kalibrieren
if (dom.GetObject("Pellets_Cal_Auto").Value())
{
if (counter_cal > 400)
{ cal = counter_cal; }
pellets_kg_ratio = cal /kg_zwbeh / cal;
}
counter_cal = 0;
Pellets_Brennerstatus
Pellets_RA-Dauer_erwartet
CuxD StateMon
RA-TIME_OFF-Event
Pellets_RES1_kg_ratio
Pellets_maxPwr
Taste (x)
Pellets_kg_letzter_Monat
Pellets_kg_letztes_Jahr
Pellets_kg_heute
Pellets_Lagerveränderung
Pellets_Lager_t
Pellets_kg_gestern
Pellets_kg_Avg10
Pellets_Cal_dev
Pellets_Day_Avg10_List
Pellets_kWh_gestern
Pellets_Aschebox_kg
Pellets_Aschebox_Fuellgrad
Pellets_RES1_gestern
Pellets_Aschebox_voll
Taste (-)
Peleo-RA-beendet
Füllgrad auf 100% setzen
fuellgrad_zwischenbeh =100;
Austragungsdauer umkopieren
var ra_time_on = (datapoints.Get
("CuxD.CUX0001001:1.TIME_ON").).Value();
Abweichung der RA-Dauer berechnen
if (ra_dauer_erwartet == 0)
{ra_dauer_erwartet = 0.0167;} // DIV/0 vermeiden
ra_dauer_abweichung = (ra_time_on - ra_dauer_erwartet) / ra_dauer_erwartet;
ra_dauer_abweichung = ra_dauer_abweichung *100;
Pellets_RA-Dauer_Abweichung
Pellets_RA-Dauer_letzte
Peleo_Lager_zubuchen
Lager zubuchen
pellets_lager_t = pellets_lager_t +
(pellets_lagerveränderung / 1000);
if (pellets_lager_t > pellets_lagermax_t)
{
pellets_lager_t = pellets_lagermax_t;
}
Lagerfüllgrad berechnen
pellets_lagerfuellgrad = (pellets_lager_t /
pellets_lagermax_t) *1000;
Lager Schwellwert prüfen
if (((pellets_lager_t * 1000) > pellets_lager_schwelle) &&
(pellets_lager_niedrig))
{
pellets_lager_niedrig = 0;
dom.GetObject("Pellets_Lager_niedrig").State(pellets_lager_niedrig);
}
Lagerreichweitenprognose berechnen
lagerreichweite = 0;
if (kg_Avg10 > 0)
{
lagerreichweite = (pellets_lager_t *1000) / kg_Avg10;
}
lagerreichweite = lagerreichweite.ToInteger();
Peleo_Lager_abbuchen
Lager abbuchen
pellets_lager_t = pellets_lager_t -
(pellets_lagerveränderung / 1000);
if (pellets_lager_t <0)
{
pellets_lager_t = 0;
}
Lager Schwellwert prüfen
if (((pellets_lager_t * 1000) < pellets_lager_schwelle) &&
(!pellets_lager_niedrig))
{
pellets_lager_niedrig = 1;
dom.GetObject("Pellets_Lager_niedrig").State(pellets_lager_niedrig);
}
Lagerfüllgrad berechnen
pellets_lagerfuellgrad = (pellets_lager_t /
pellets_lagermax_t) *1000;
Lagerreichweitenprognose berechnen
lagerreichweite = 0;
if (kg_Avg10 > 0)
{
lagerreichweite = (pellets_lager_t *1000) / kg_Avg10;
}
lagerreichweite = lagerreichweite.ToInteger();
Pellets_Lagerfuellgrad
Pellets_Lager_niedrig
Pellets_Lagerreichweite