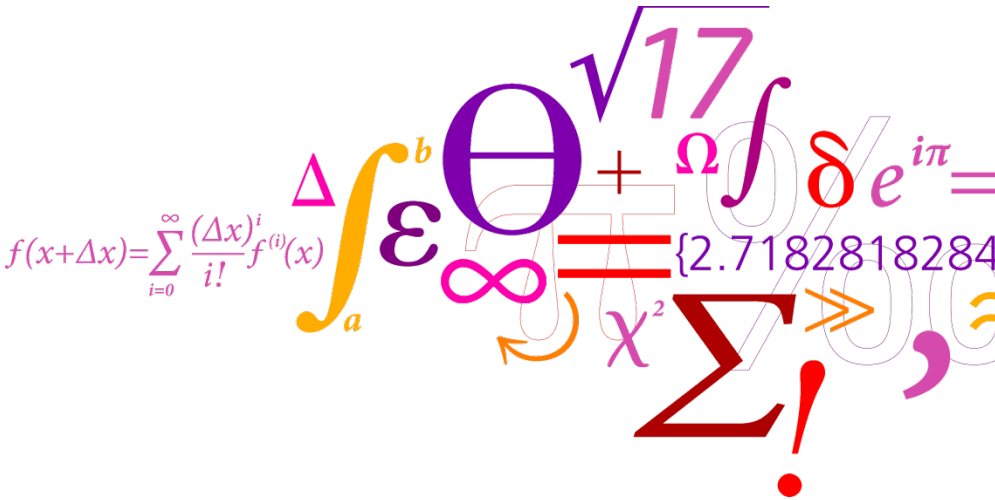
Reguleringsteknik 1

J. Christian Andersen

Kursusuge 8

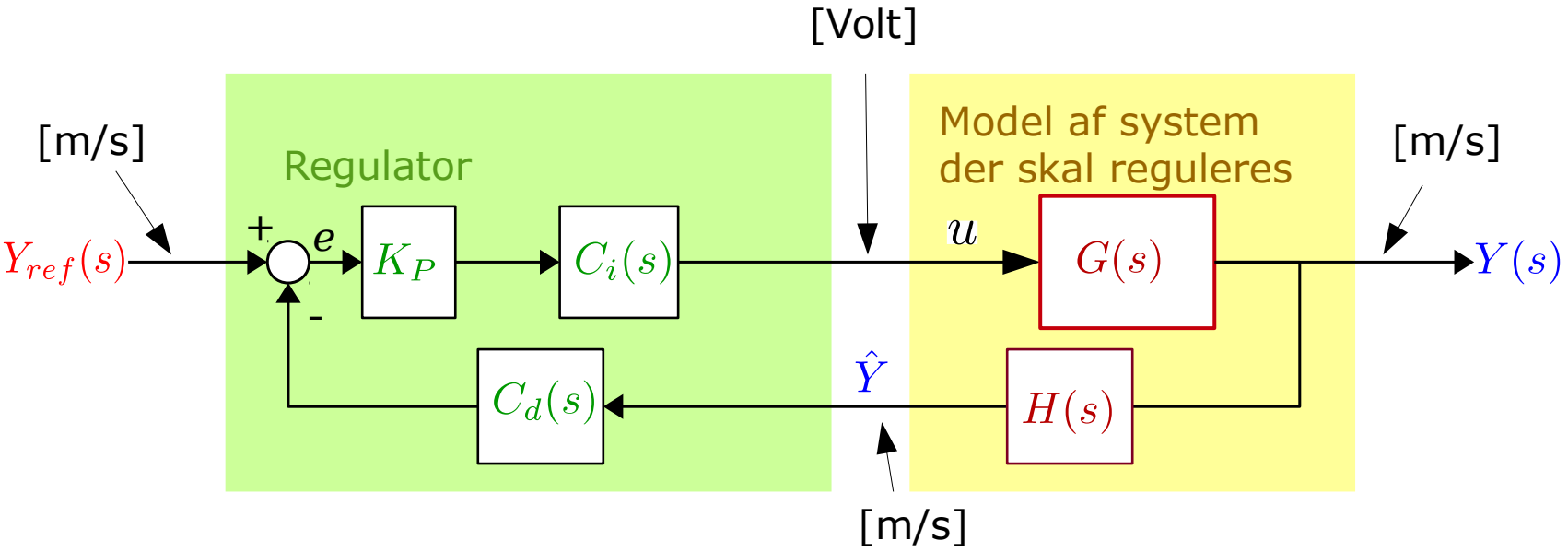
Plan

- Implementering af regulator
  - Andet format (f.eks. MATLAB)
  - Analog implementering
  - Digital implementering (ikke pensum)



DTU Electrical Engineering
Department of Electrical Engineering

Implementering



Resultat af PI-Lead regulator design:

$$\tau_i = 0.05,\ \tau_d = 0.05,\ \alpha = 0.8\ K_P = 12$$

$$C_i = \frac{\tau_i s + 1}{\tau_i s}$$

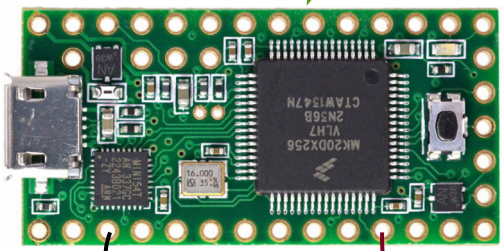
$$C_d = \frac{\tau_d s + 1}{\alpha \tau_d s + 1}$$

Implementering
- virkeligt system
- lille skala

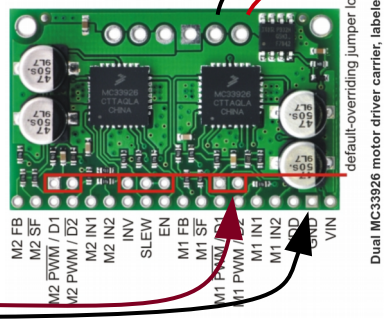
9-14.3V

48 pulser per rev

C++

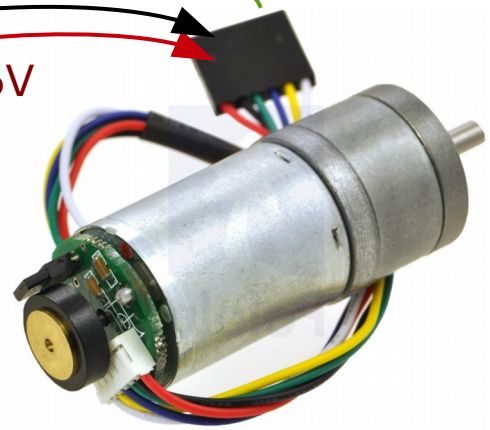


Ref:
[m/s]



PWM

0-6V



[Volt]

[m/s]

Regulator

Model af system
der skal reguleres

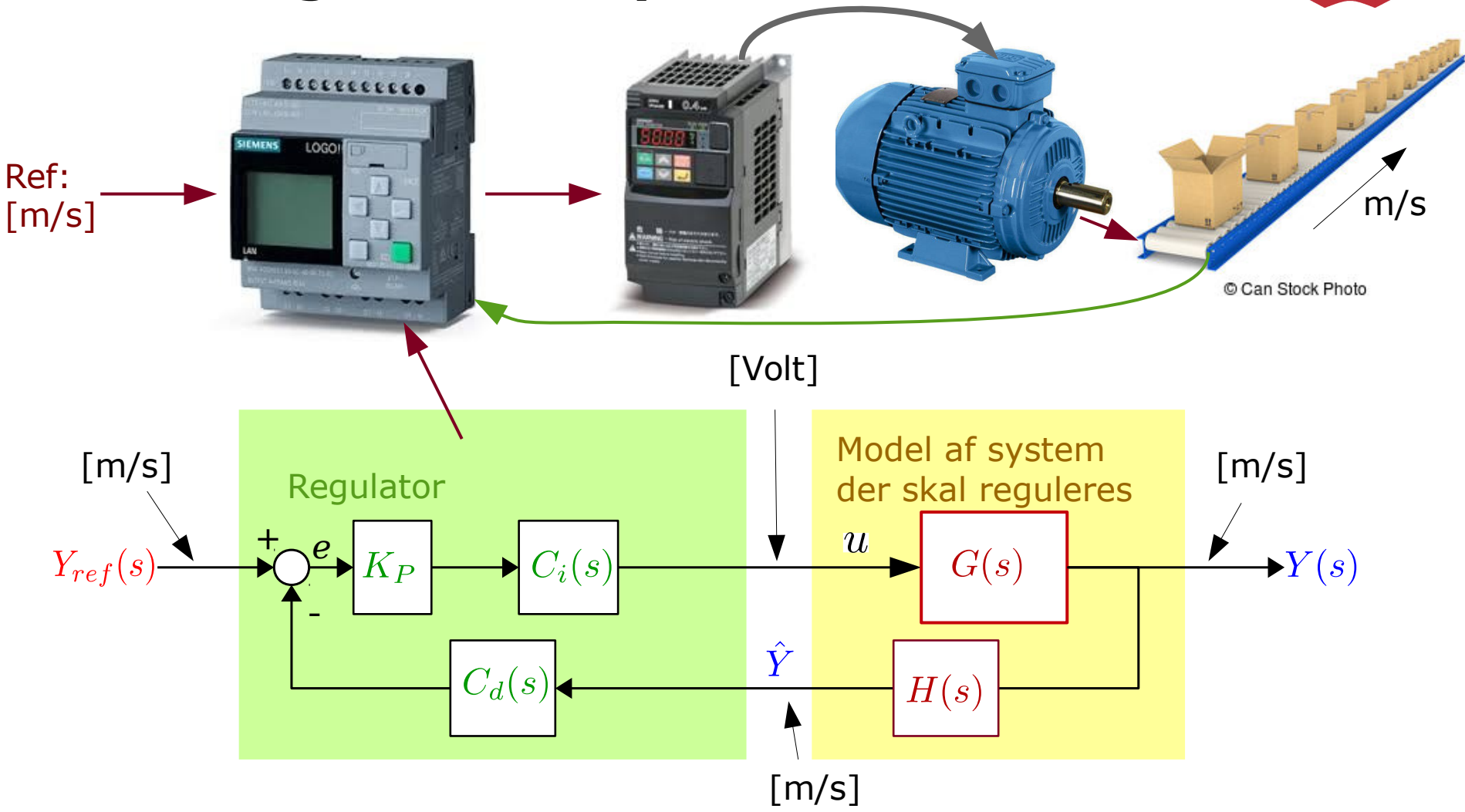
[m/s]

$Y_{ref}(s)$ $+$ e $-$ K_P $C_i(s)$ u $G(s)$ $Y(s)$

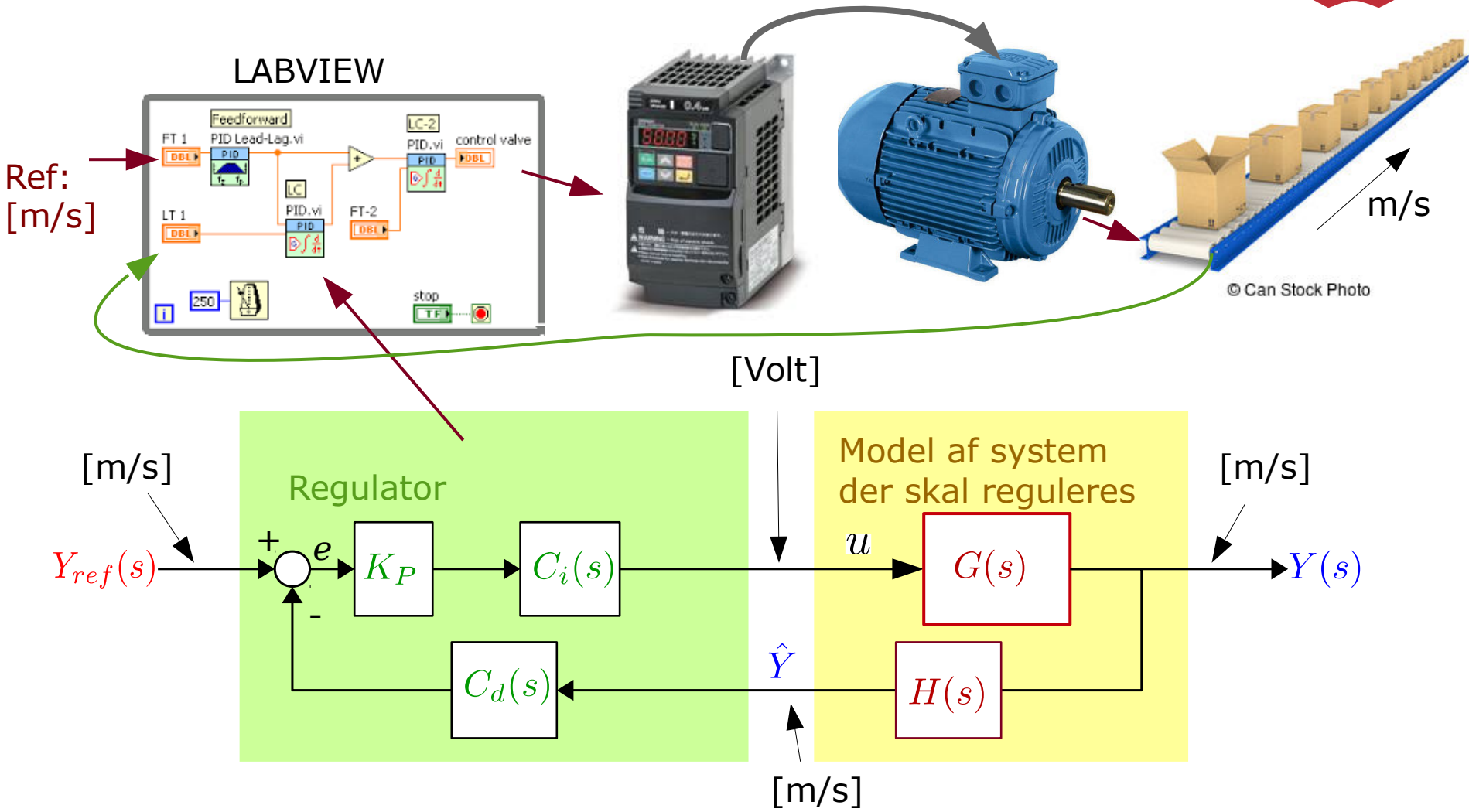
$C_d(s)$ $\hat{Y}$ $H(s)$

[m/s]

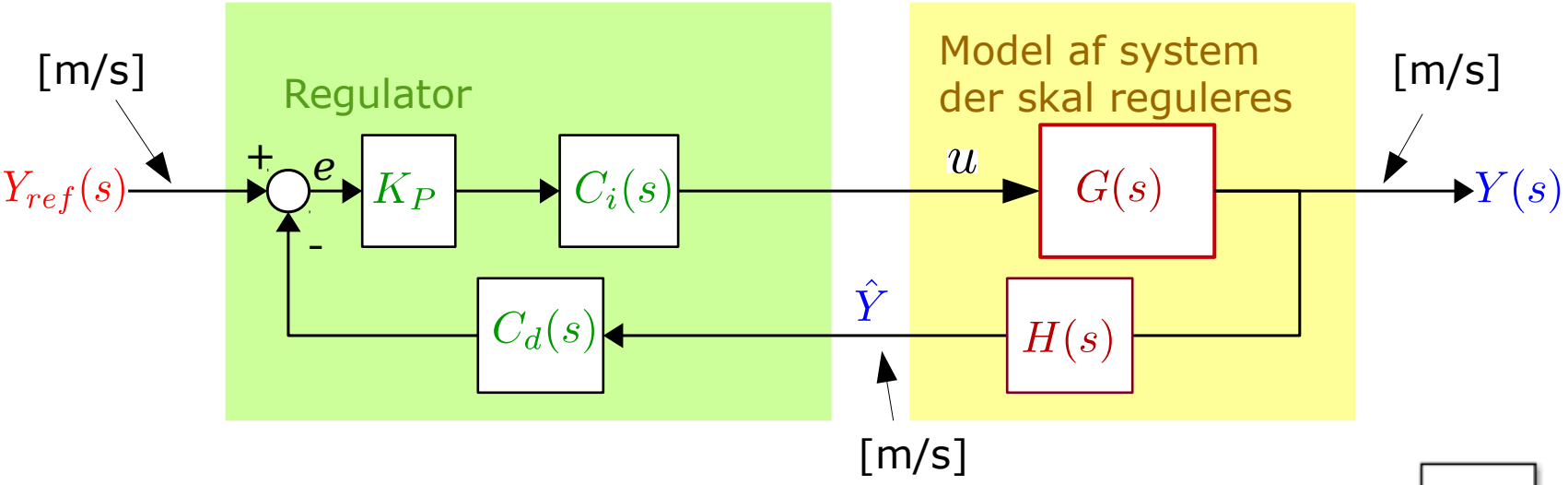
Implementering
- virkeligt industri system



Implementering - Labview



Implementering
- PID blok til rådighed (MATLAB og andre)



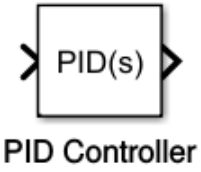
Design

$$\tau_i = 0.05,\ \tau_d = 0.05,\ \alpha = 0.8\ K_P = 12$$

$$C(s) = K_P\left(1 + \frac{1}{\tau_i s}\right)\frac{\tau_d s + 1}{\alpha\tau_d s + 1}$$

Implementer med

MATLAB PID-blok
(parallel form)



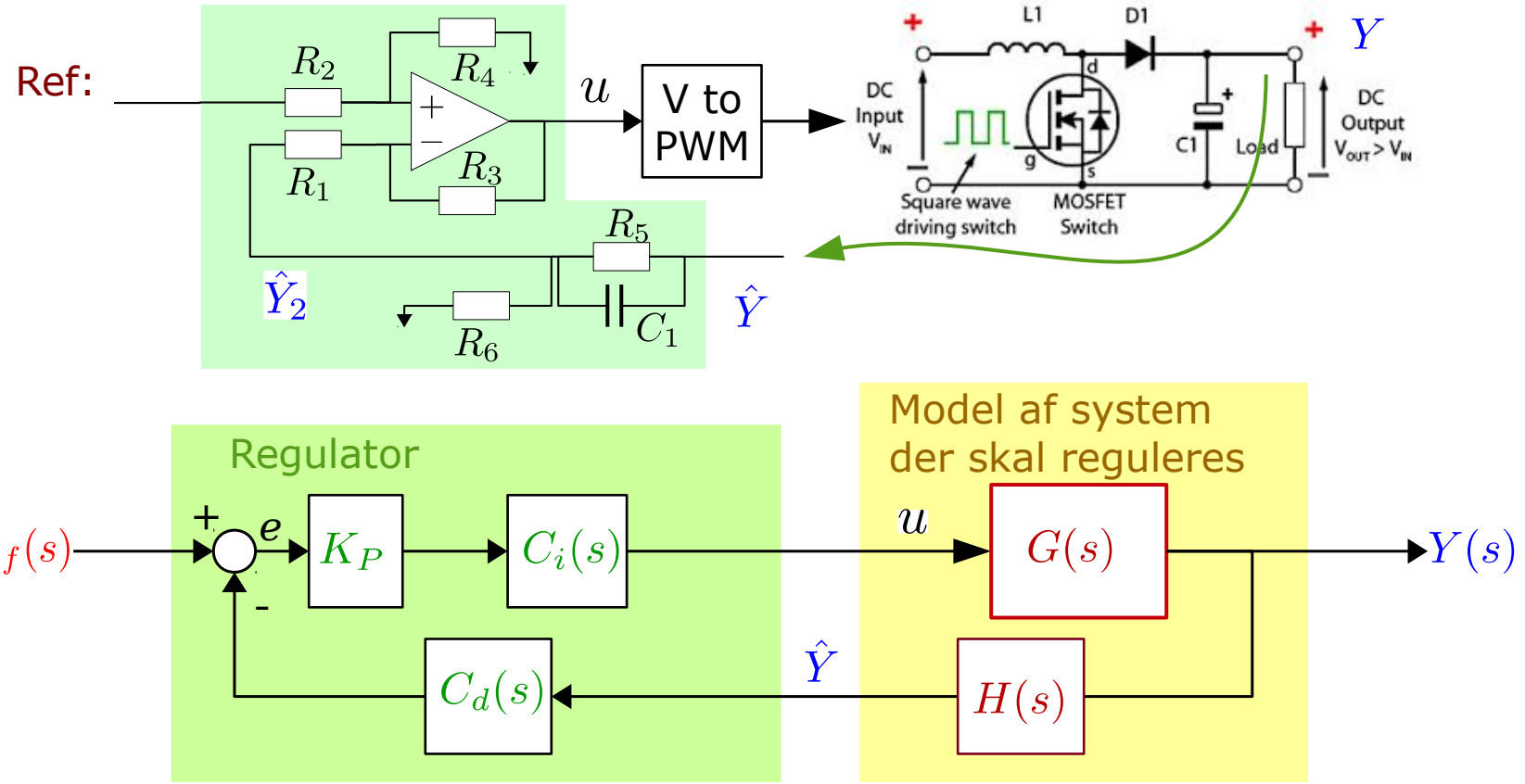
$$C(s) = P + I\frac{1}{s} + D\frac{N}{1 + N\frac{1}{s}}$$

$$P = K_P \qquad D \approx \tau_d K_P$$

$$I = \frac{K_P}{\tau_i} \qquad N = \frac{1}{\alpha\tau_d}$$

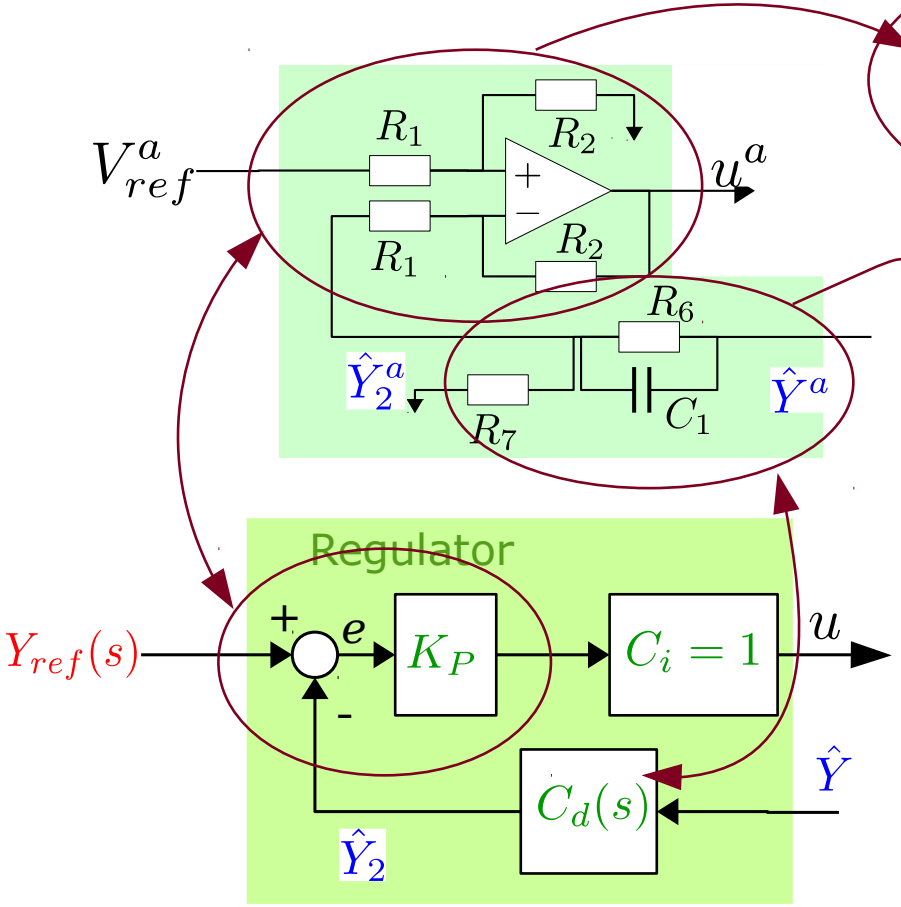
PS: Lead i fremadgren

Implementering
- virkeligt system
- analogt



Implementering
- virkeligt system
- analogt

Se også bog afsnit 6.6.

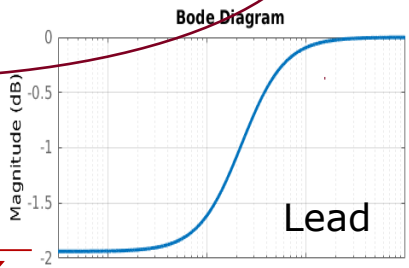


Differensforstærker:

$$u^a = (V^a_{ref} - \hat{Y}^a_2)\frac{R_2}{R_1}$$

$$K_P = \frac{R_2}{R_1}$$

$$u = (Y_{ref} - \hat{Y}_2)K_P$$



Spændingsdeler:

$$C^a_d = \frac{\hat{Y}^a_2}{\hat{Y}^a} = \frac{R_7}{R_7 + R_6 \parallel Z_C}$$

$$C^a_d = \underbrace{\frac{R_7}{R_6 + R_7}}_{K_H} \frac{R_6 C s + 1}{\frac{R_7}{R6+R_7} R_6 C s + 1}$$

$$C_d = \frac{\tau_d s + 1}{\alpha \tau_d s + 1}$$

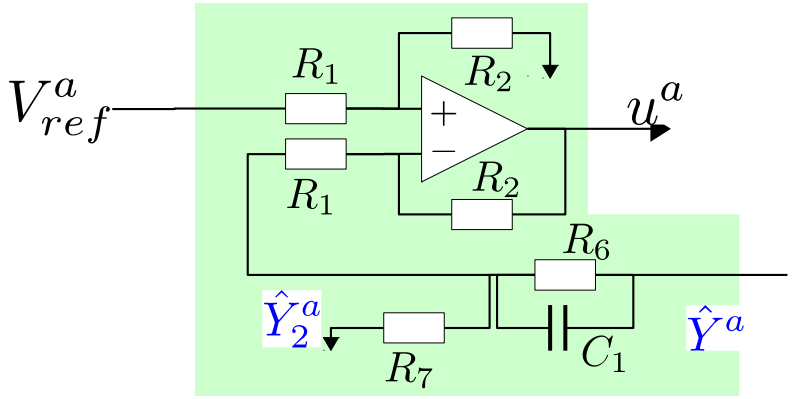
$$\tau_d = R_6 C$$

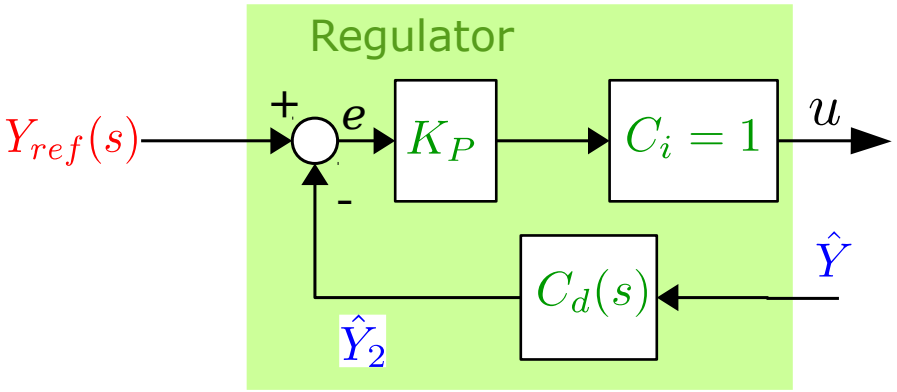
$$\alpha = \frac{R_7}{R_6 + R_7}$$

Da der i åben sløjfe er ganget med K_H for meget må der kompenseres:

$$\frac{K_P}{K_H} = \frac{R_2}{R_1}$$

Implementering
- virkeligt system
- analogt





$$K_H = \frac{R_7}{R_6 + R_7}$$

$$\tau_d = R_6 C$$

$$\alpha = \frac{R_7}{R_6 + R_7}$$

$$\frac{K_P}{K_H} = \frac{R_2}{R_1}$$

$$\tau_i = 0.05,\ \tau_d = 0.05,\ \alpha = 0.8\ K_P = 12$$

Isolation af led: $R_7 << R_1$

Valg: $\underline{R_7 = 1\text{k}\Omega},\ \underline{R_1 = 47\text{k}\Omega}$

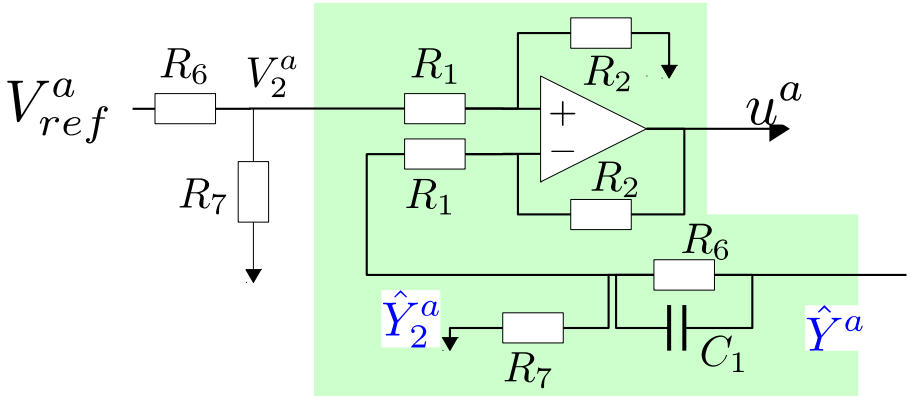
$$\Rightarrow R_6 = \frac{R_7}{\alpha} - R_7 \Rightarrow \underline{R_6 = 250\Omega}$$

$$\Rightarrow C = \frac{\tau_d}{R_6} \Rightarrow \underline{C = 200\mu\text{F}}$$

$$\Rightarrow R_2 = R_1 K_P \frac{1}{K_H} \Rightarrow \underline{R_2 = 705\text{k}\Omega}$$

Effekt af K_H på V_{ref}^a ?

Implementering
- virkeligt system
- analogt



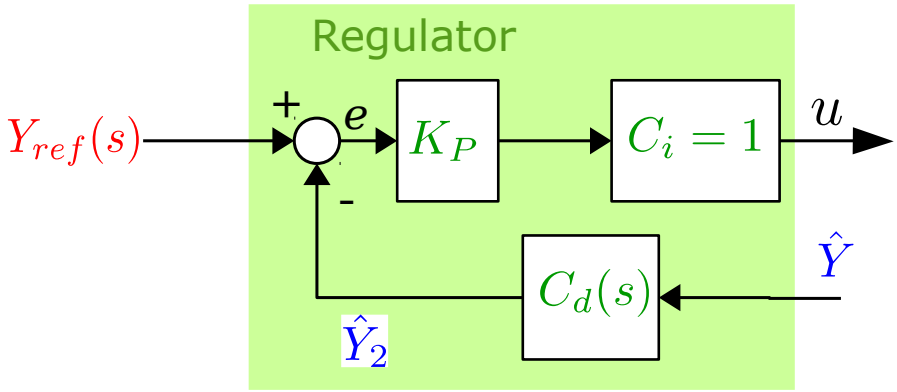
$$K_H = \frac{R_7}{R_6 + R_7}$$

$$\tau_i = 0.05,\ \tau_d = 0.05,\ \alpha = 0.8\ K_P = 12$$

Effekt af K_H på V^a_{ref} ?

$$C^a_{d,ss} = \frac{\hat{Y}^a_{2,ss}}{\hat{Y}^a_{ss}} = K_H$$

En tilsvarende spændingsdeler på reference vil kompensere (eller en lavere Vref).

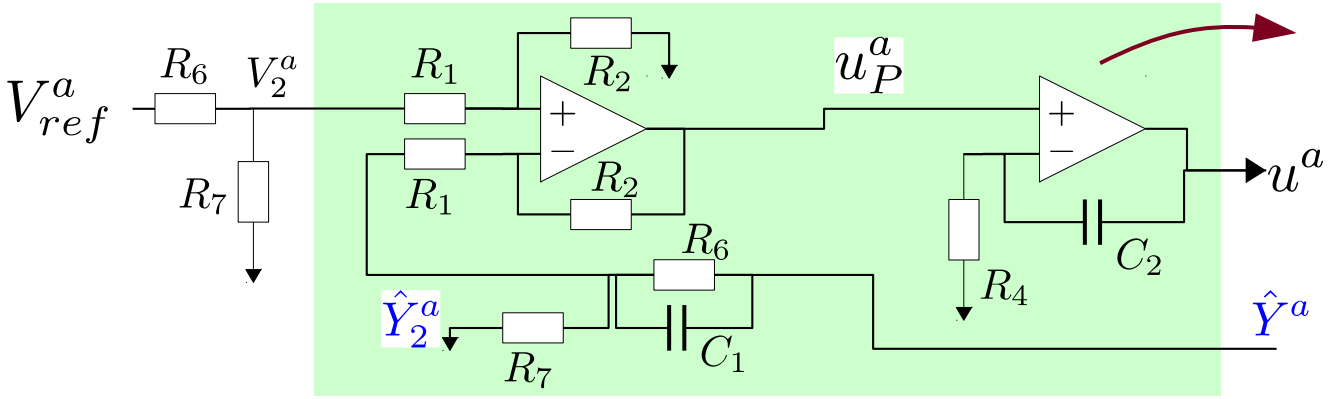


Hvad med τ_i ?

Implementering
- virkeligt system
- analogt

Hvad med τ_i ?

$\tau_i = 0.05,\ \tau_d = 0.05,\ \alpha = 0.8\ K_P = 12$

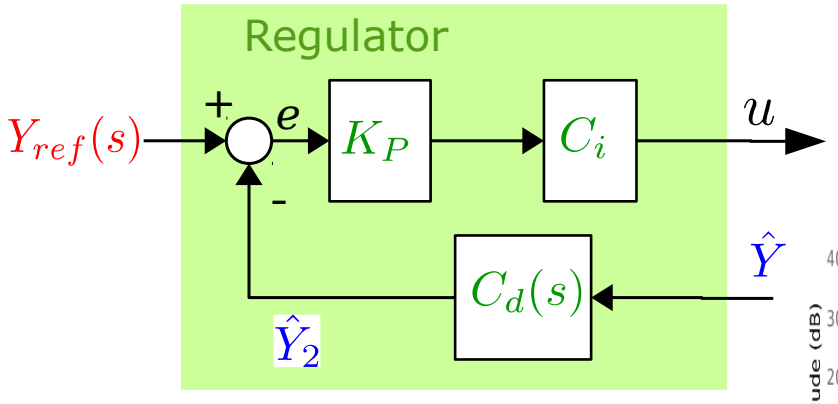


Ikke inverterende op-amp:

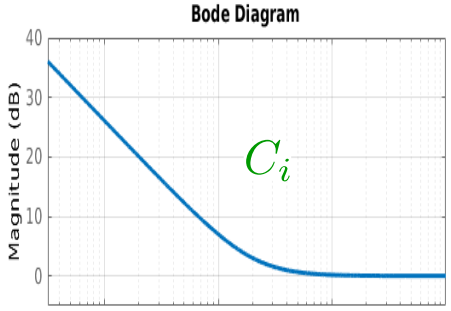
$$C_i^a = \frac{u^a}{u_P^a} = 1 + \frac{Z_{C2}}{R_4}$$

$$C_i^a = 1 + \frac{1}{R_4 C_2 s}$$

$$\tau_i = R_4 C_2$$



$$C_i = 1 + \frac{1}{\tau_i s}$$



Valg: $C_2 = 100\text{nF}$

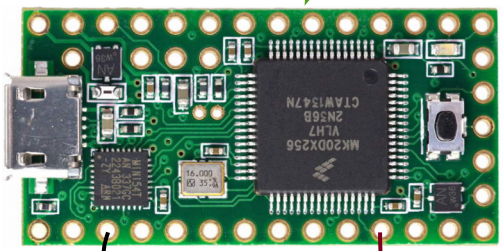
$$R_4 = \frac{\tau_i}{C_2} \Rightarrow R_4 = 50\text{k}\Omega$$

Implementering
- virkeligt system
- digitalt

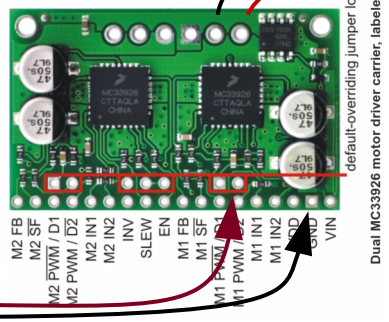
9-14.3V

48 pulser per rev

C++

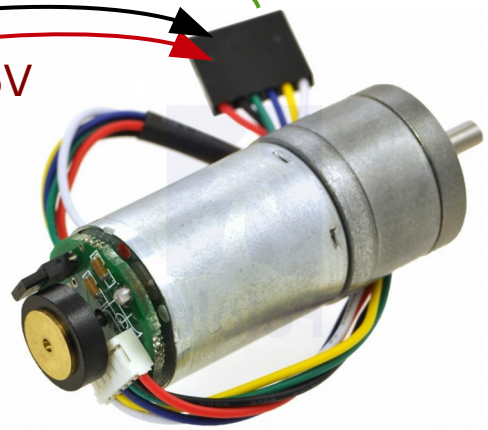


Ref:
[m/s]

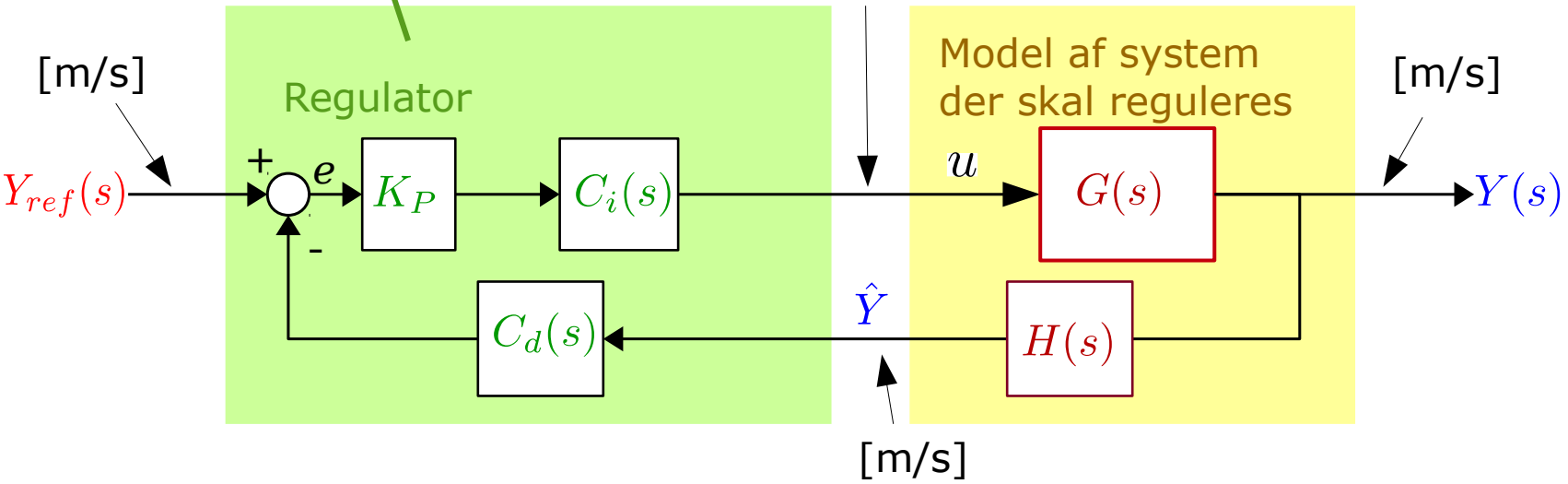


PWM

0-6V

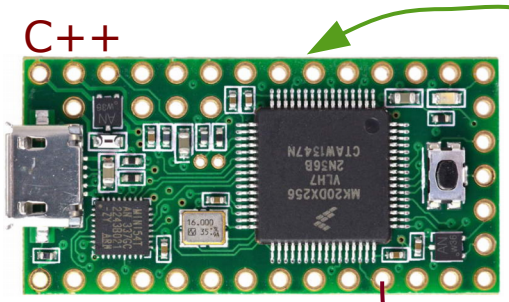


[Volt]



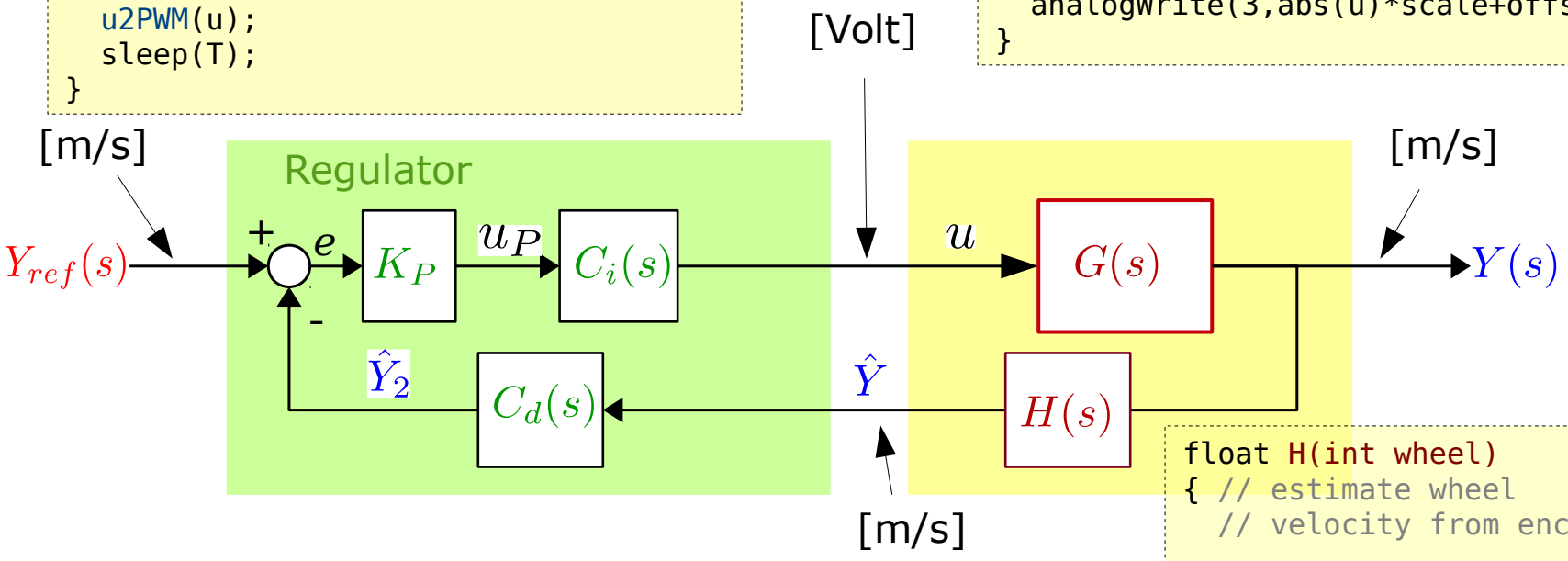
Implementering
- virkeligt system
- digitalt

$\tau_i = 0.05,\ \tau_d = 0.05,\ \alpha = 0.8\ K_P = 12$




```
void loop()
{ // runs every T seconds
  float Y2 = lead(H(1));
  float e = Yref - Y2;
  float up = e * Kp;
  float u = integ(up);
  u2PWM(u);
  sleep(T);
}
```

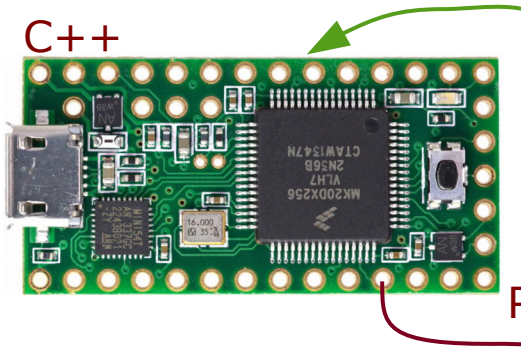
```
void u2PWM(float u)
{ // u in motor volts
  if (u > 0)
    digitalWrite(2, HIGH);
  else
    digitalWrite(2, LOW);
  analogWrite(3,abs(u)*scale+offset);
}
```



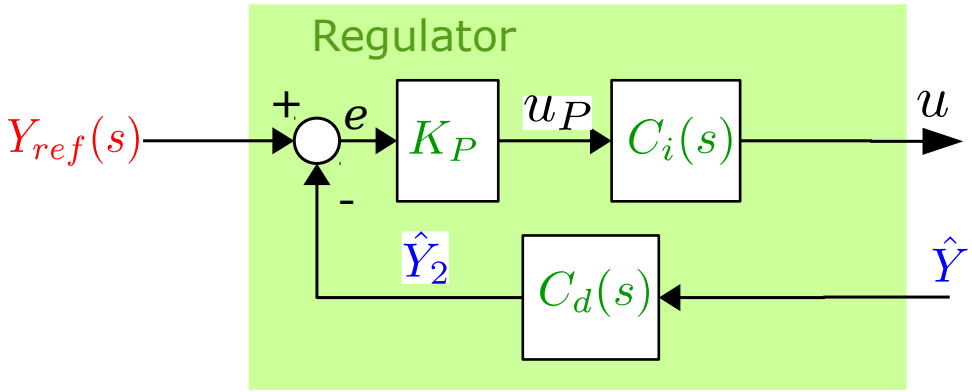

```
float H(int wheel)
{ // estimate wheel
  // velocity from encoder
  ...
  return velocityEst;
}
```

Implementering
- virkeligt system
- digitalt

$\tau_i = 0.05,\ \tau_d = 0.05,\ \alpha = 0.8,\ K_P = 12$




```
void loop()
{ // runs every T seconds
  float Y2 = lead(H(1));
  float e = Yref - Y2;
  float up = e * Kp;
  float u = integ(up);
  u2PWM(u);
  sleep(T);
}
```



$$C_d = \frac{\tau_d s + 1}{\alpha \tau_d s + 1}$$

- Hver T sekunder? hvilken T?
- Hvordan omsættes C_i og C_d til C++?

T vælges (betydeligt) hurtigere end hurtigste tidskonstant.
- her er hurtigste tidskonstant polen i Lead led:

$$\tau_{d2} = \alpha \tau_d = 0.04\ (40\ \text{ms})$$

Der kunne f.eks vælges:
T=10ms, eller bedre T=1ms
Effekten kan medregnes i bodeplot.

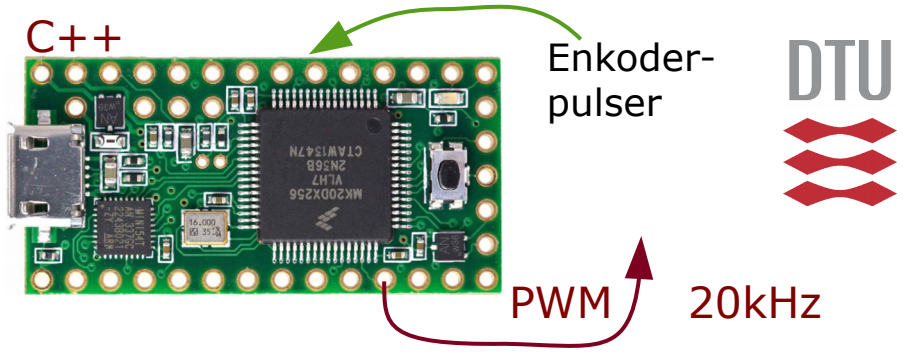
G(s) kan omsættes til G(z) med

$$s = \frac{2(z-1)}{T(z+1)}$$

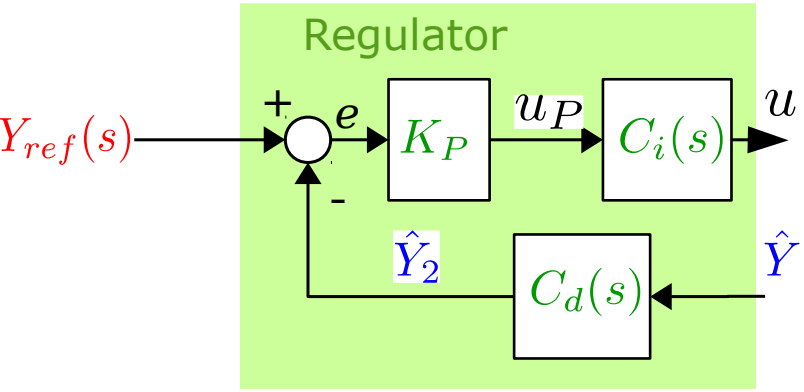
som lært i digital signalbehandling - eller intro i supplerende video.

Implementering - virkeligt system - digitalt

$\tau_i = 0.05,\ \tau_d = 0.05,\ \alpha = 0.8,\ K_P = 12$




```
void loop()
{ // runs every T seconds
  float Y2 = lead(H(1));
  float e = Yref - Y2;
  float up = e * Kp;
  float u = integ(up);
  u2PWM(u);
  sleep(T);
}
```



Lead i C++:

$$C_d = \frac{\tau_d s + 1}{\alpha\tau_d s + 1} \qquad s = \frac{2(z-1)}{T(z+1)}$$

$$C_d(z) = \frac{(T + 2\tau_d)z + T - 2\tau_d}{(2\alpha\tau + T)z - 2\alpha\tau_d + T}$$

Z^{-1} er en tidsforsinkelse på netop 1 T, så

$$C_d(z) = \frac{Y_2}{Y} = \frac{T + 2\tau_d + (T - 2\tau_d)z^{-1}}{2\alpha\tau + T + (-2\alpha\tau_d + T)z^{-1}}$$

$$Y_2(2\alpha\tau_d + T + (-2\alpha\tau_d + T)z^{-1}) = Y(T + 2\tau_d + (T - 2\tau_d)z^{-1})$$

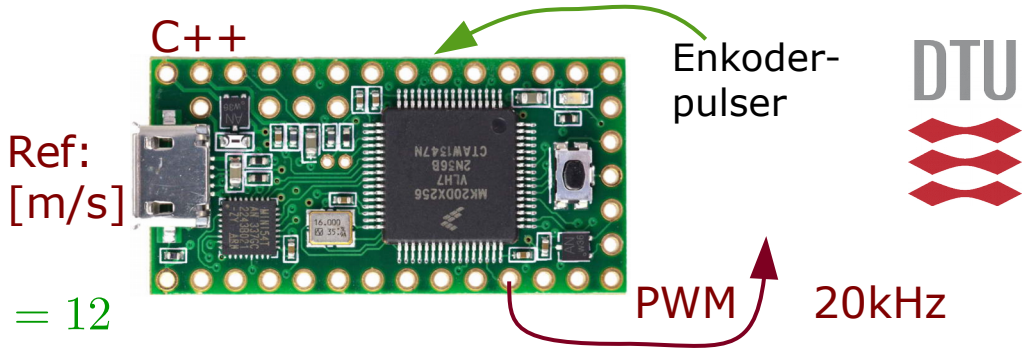
$$Y_2(2\alpha\tau_d + T) = (T + 2\tau_d)Y + (T - 2\tau_d)Yz^{-1} - (-2\alpha\tau_d + T)Y_2 z^{-1}$$

Eller Yz^{-1} er Y fra sidste udregning Y_{k-1}

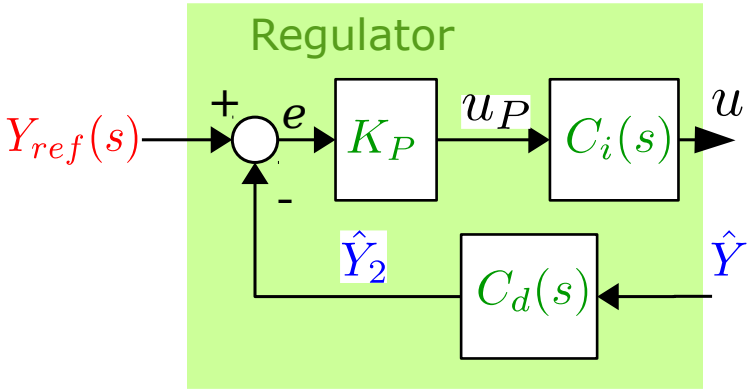
$$Y2_k = \frac{1}{2\alpha\tau_d + T}\left((T + 2\tau_d)Y_k + (T - 2\tau_d)Y_{k-1} + (2\alpha\tau_d - T)Y2_{k-1}\right)$$

Implementering - virkeligt system - digitalt

$\tau_i = 0.05,\ \tau_d = 0.05,\ \alpha = 0.8,\ K_P = 12$




```
void loop()
{ // runs every T seconds
  float Y2 = lead(H(1));
  float e = Yref - Y2;
  float up = e * Kp;
  float u = integ(up);
  u2PWM(u);
  sleep(T);
}
```



$$C_d = \frac{\tau_d s + 1}{\alpha \tau_d s + 1} \qquad s = \frac{2(z-1)}{T(z+1)}$$

$$Y2_k = \frac{1}{2\alpha\tau_d + T}\left((T + 2\tau_d)Y_k + (T - 2\tau_d)Y_{k-1} + (2\alpha\tau_d - T)Y2_{k-1}\right)$$

Lead i C++:

```
float Y=0, Y2=0;
float k1 = (T+2*taud)/(2*alpha*taud+T);
float k2 = (T-2*taud)/(2*alpha*taud+T);
float k3 = (2*alpha*taud-T)/(2*alpha*taud+T);

float lead(float Yny)
{ // implement Lead
  Y2 = k1*Yny + k2*Y + k3*Y2;
  Y = Yny;
  return Y2;
}
```