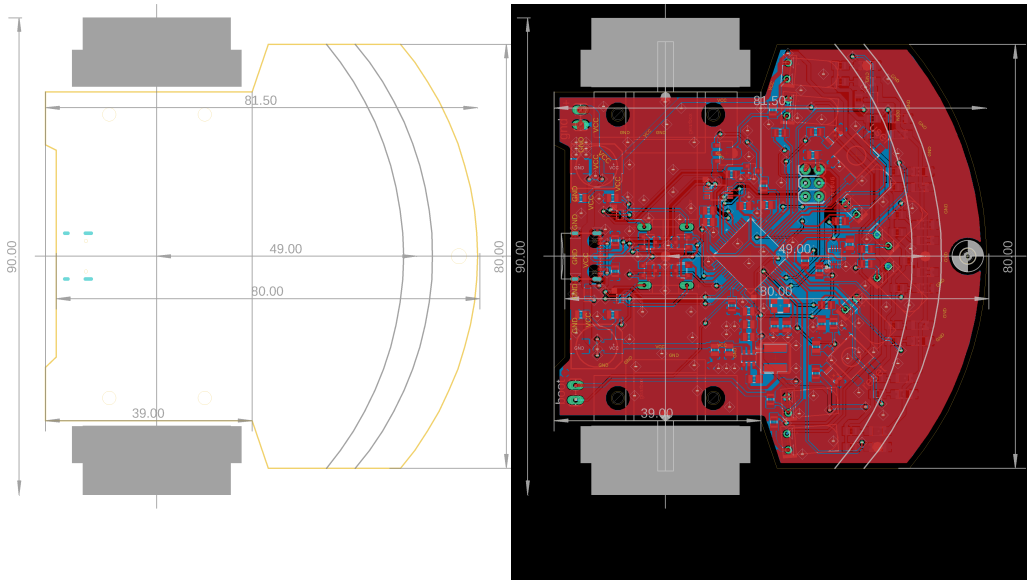
Chapter 1

Hardware description



(a) mechanical dimensions (b) overall PCB layout

Figure 1.1: Robot PCB

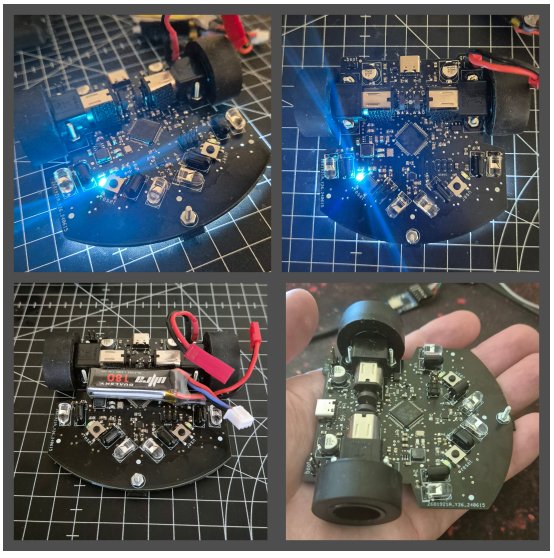


Figure 1.2: Robot photo

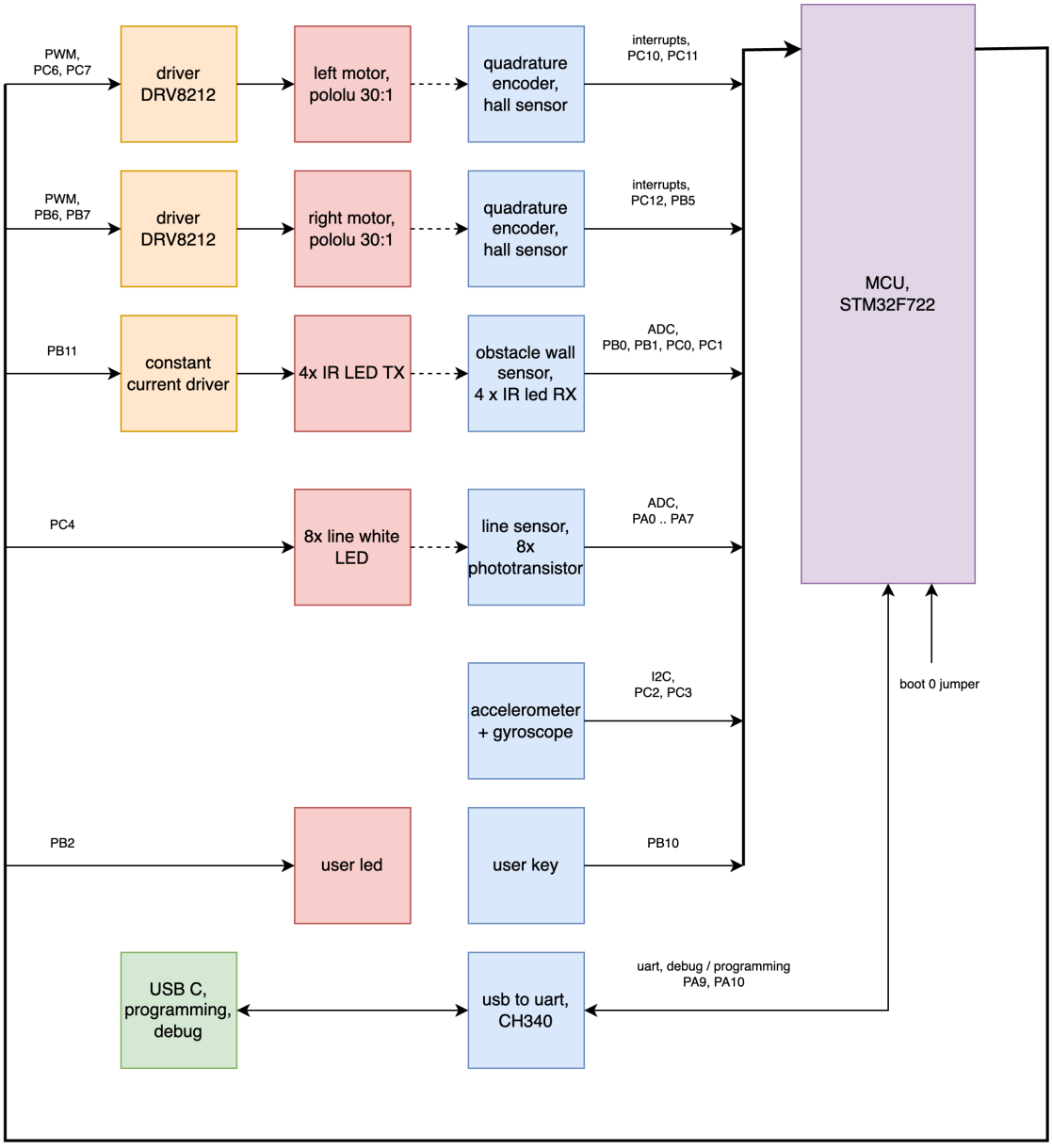


Figure 1.3: block diagram of DC motor version

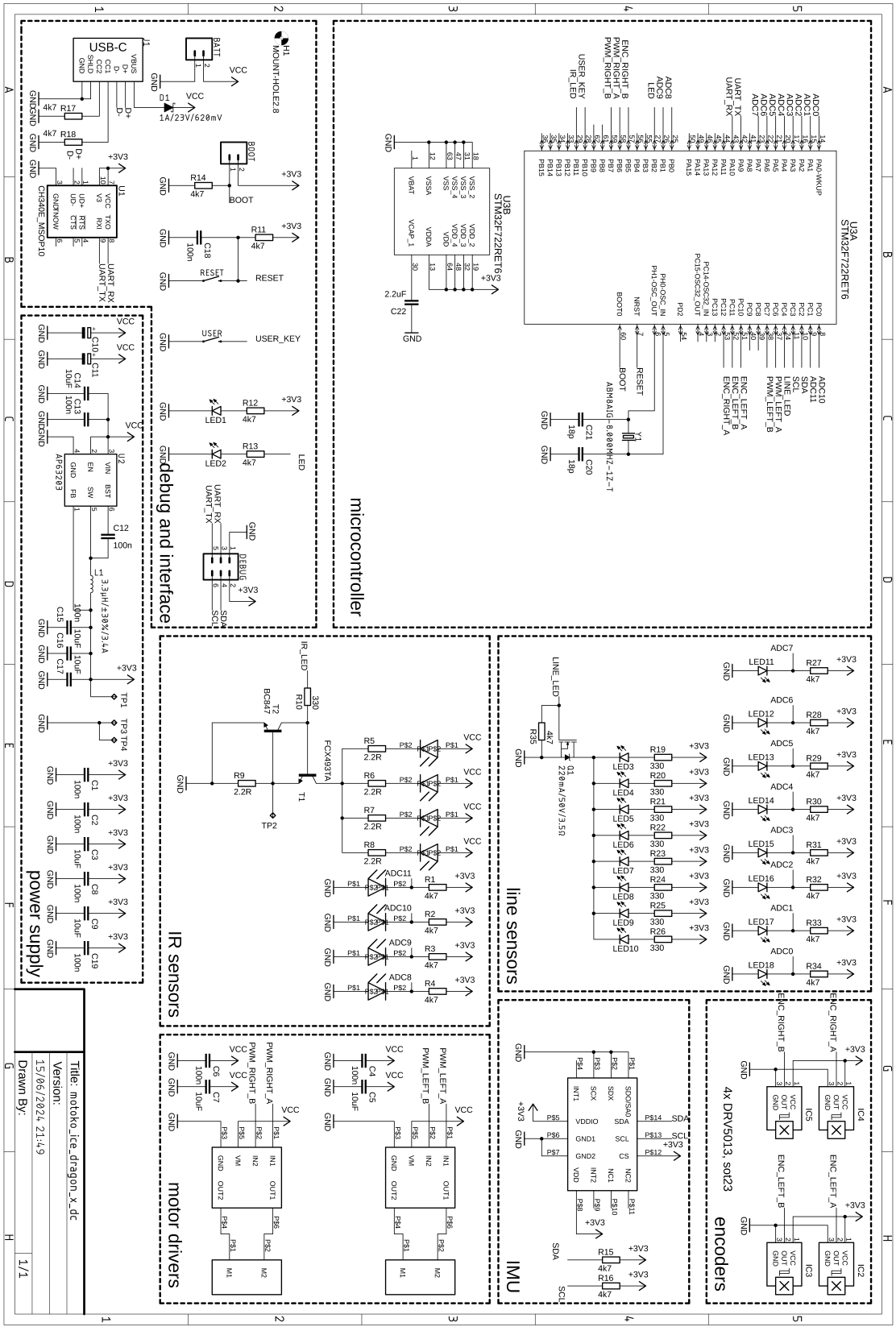


Figure 1.4: schematic diagram of DC motor version

pin number	pin name	function	peripheral	AF number
14	PA0	line sensor ADC0, right	ADC1	analog in
15	PA1	line sensor ADC1	ADC1	analog in
16	PA2	line sensor ADC2	ADC1	analog in
17	PA3	line sensor ADC3	ADC1	analog in
20	PA4	line sensor ADC4	ADC1	analog in
21	PA5	line sensor ADC5	ADC1	analog in
22	PA6	line sensor ADC6	ADC1	analog in
23	PA7	line sensor ADC7, left	ADC1	analog in
25	PB0	IR sensor ADC8, front left	ADC1	analog in
26	PB1	IR sensor ADC9, left	ADC1	analog in
8	PC0	IR sensor ADC10, right	ADC1	analog in
9	PC1	IR sensor ADC11, front right	ADC1	analog in
51	PC10	encoder left A	GPIOC	
52	PC11	encoder left B	GPIOC	
53	PC12	encoder right A	GPIOC	
57	PB5	encoder right B	GPIOB	
37	PC6	PWM left A	TIM3_CH1	AF2
38	PC7	PWM left B	TIM3_CH2	AF2
58	PB6	PWM right A	TIM4_CH1	AF2
59	PB7	PWM right B	TIM4_CH2	AF2
10	PC2	IMU I2C, SDA	GPIOC	
11	PC3	IMU I2C, SCL	GPIOC	
42	PA9	uart TX	UART1	AF7
43	PA10	uart RX	UART1	AF7
27	PB2	user LED	GPIOB	
28	PB10	user Key	GPIOB	
29	PB11	IR LED control	GPIOB	
24	PC4	line LED control	GPIOC	
7	NRST	reset		
60	BOOT0	firmware bootloader control		

Table 1.1: pin mapping and function

Chapter 2

Motor control

motor velocity controller

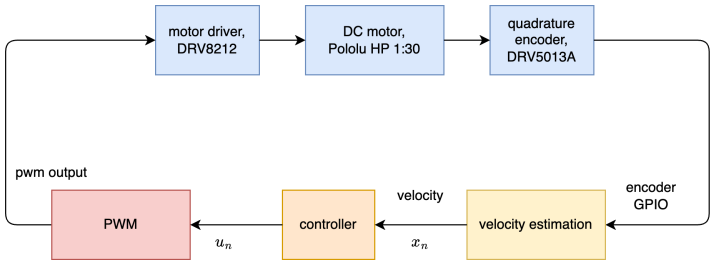


Figure 2.1: Velocity control overview

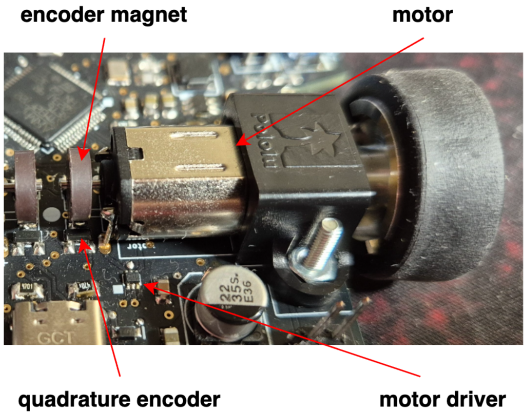


Figure 2.2: Real system photo

2.1 PID control

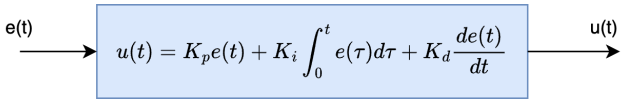


Figure 2.3: Textbook continuous PID controller

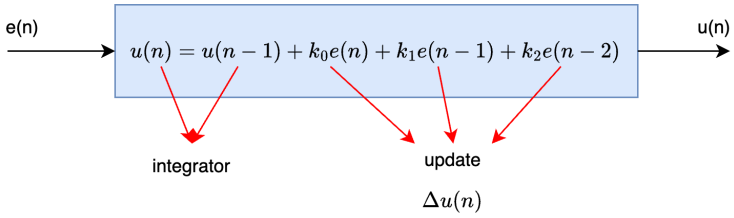


Figure 2.4: Discrete PID controller

$$k_0 = K_p + K_i \Delta t + \frac{K_d}{\Delta t}$$
$$k_1 = -K_p - 2\frac{K_d}{\Delta t}$$
$$k_2 = \frac{K_d}{\Delta t}$$

PID control - P only controller

- target value : **1000rpm**
- P-only control causes **steady state error**

$$u(n+1) = u(n-1) + k_p e(n) - k_p e(n-1)$$
$$k_p = 0.01$$

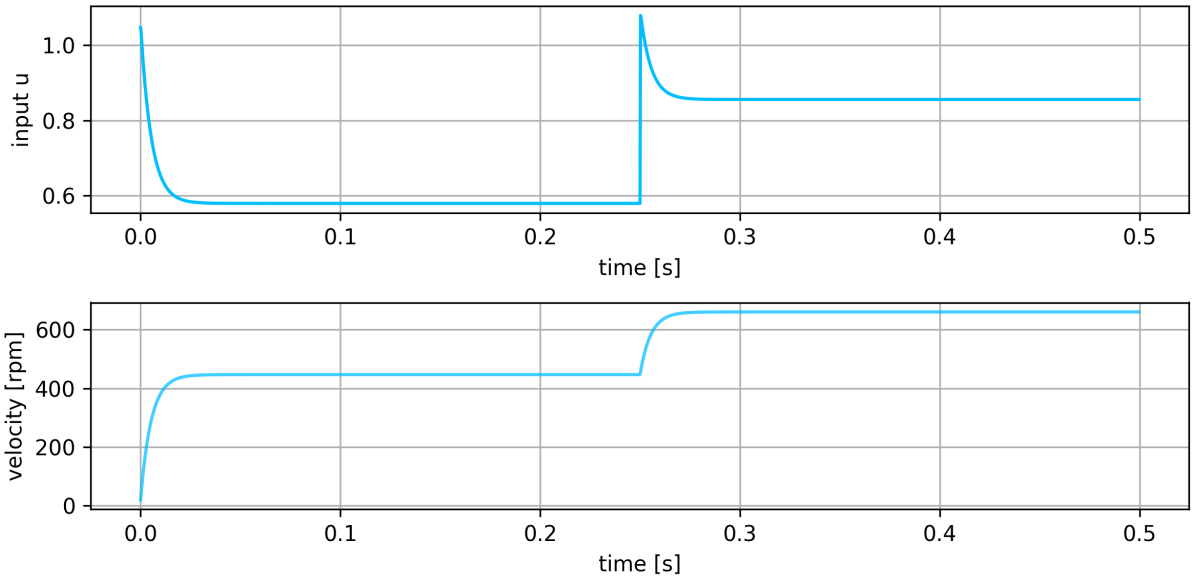


Figure 2.5: P-only controller

PID control - PI

too high I term

- target value : **1000rpm**
- PI control **removes steady state error**
- too high I term causes **oscilations and overshot**

$$u(n+1) = u(n-1) + (k_p + k_i\Delta t)e(n) - k_p e(n-1)$$
$$k_p = 0.01$$
$$k_i = 0.005$$

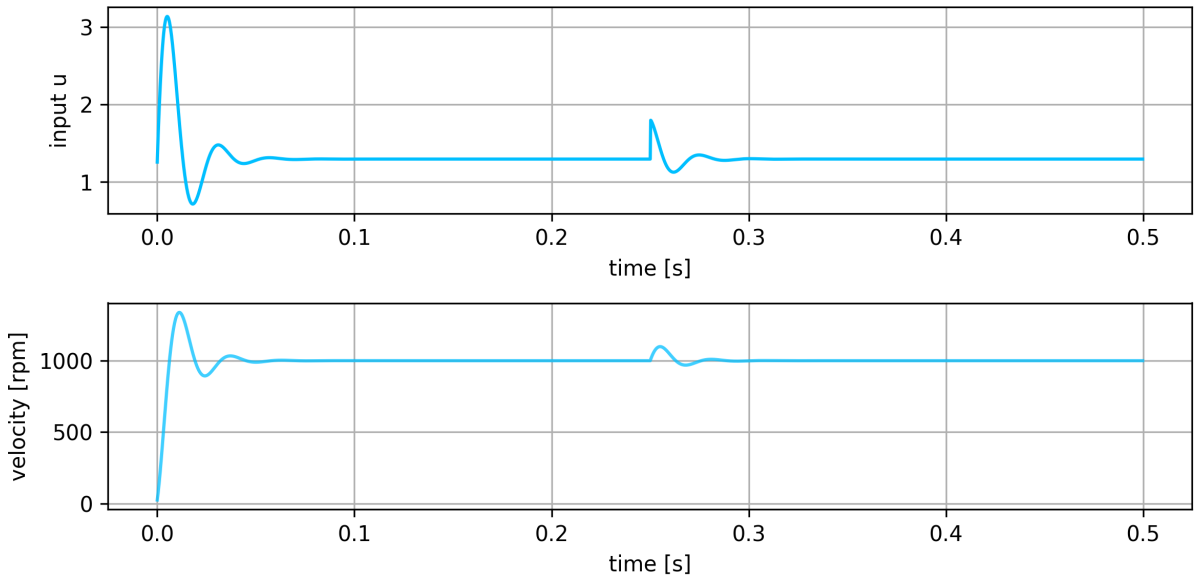


Figure 2.6: PI controller

correct I term

- target value : **1000rpm**
- correct tunned PI controller for 1st order system

- **no overshot, no steady state error**

$$u(n+1) = u(n-1) + (k_p + k_i\Delta t)e(n) - k_p e(n-1)$$
$$k_p = 0.01$$
$$k_i = 0.0002$$

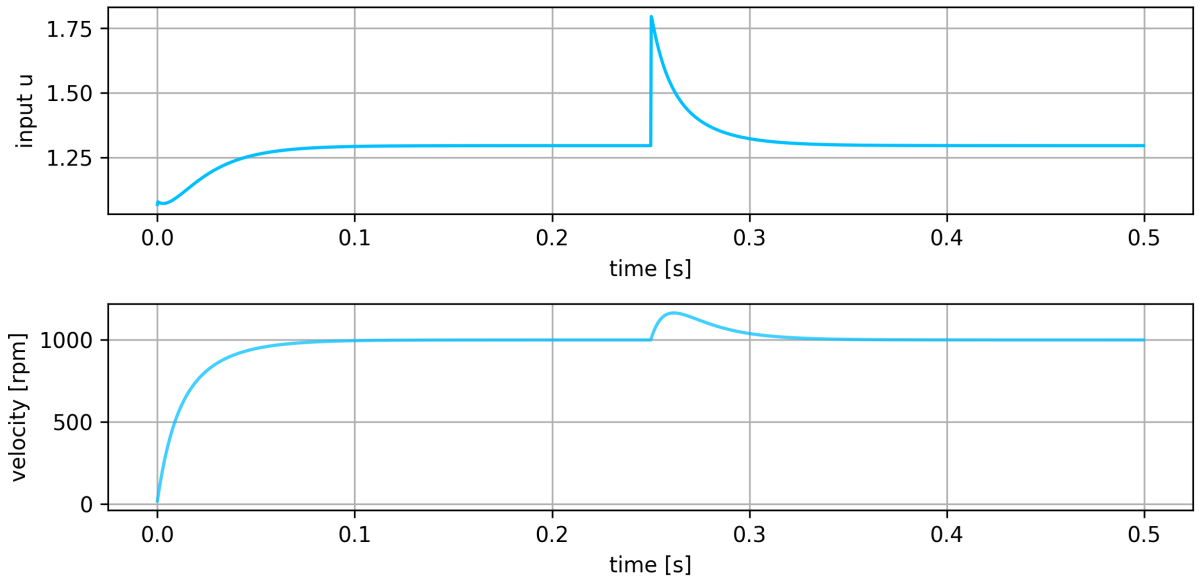


Figure 2.7: PI controller with correct parameters

complete discrete PID algorithm

1. calculate u-change candidate :

$$\Delta\hat{u}(n) = k_0 e(n) + k_1 e(n) + k_2 e(n)$$

2. clip maximum allowed u-change, to avoid u-kick :

$$\Delta u(n) = clip(\Delta\hat{u}(n), -du_{min}, du_{max})$$

3. clip maximum allowed u value, to avoid saturation / windup :

$$u(n) = clip(u(n-1) + \Delta u(n), -u_{min}, u_{max})$$

```
def __init__(self, kp, ki, kd, antiwindup = 10**10, du_max=10**10):
    self.k0 = kp + ki + kd
    self.k1 = -kp -2.0*kd
    self.k2 = kd

    self.e0 = 0.0
    self.e1 = 0.0
    self.e2 = 0.0


    self.antiwindup = antiwindup
    self.du_max     = du_max
```

Figure 2.8: Initialisation

```
def forward(self, xr, x, u_prev):
    # error compute
    self.e2 = self.e1
    self.e1 = self.e0
    self.e0 = xr - x

    du = self.k0*self.e0 + self.k1*self.e1 + self.k2*self.e2

    #kick clipping, maximum output value change limitation
    du  = numpy.clip(du, -self.du_max , self.du_max)

    #entiwindup, maximum output value limitation
    u   = numpy.clip(u_prev + du,  -self.antiwindup, self.antiwindup)

    return u
```

Figure 2.9: Main loop