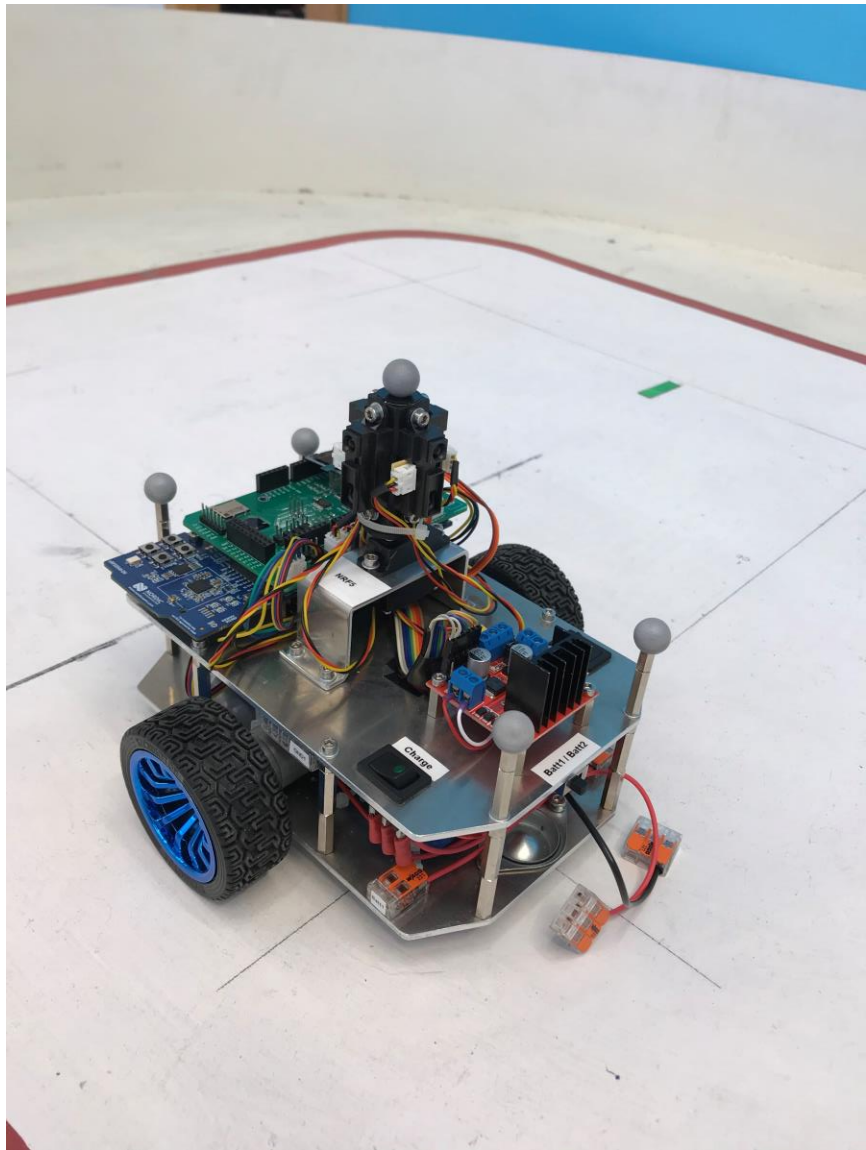
Software for DC Motor Speed Controller Card for Differential Drive Robot

Thomas Andersen - TTK8 prosjekt 2021

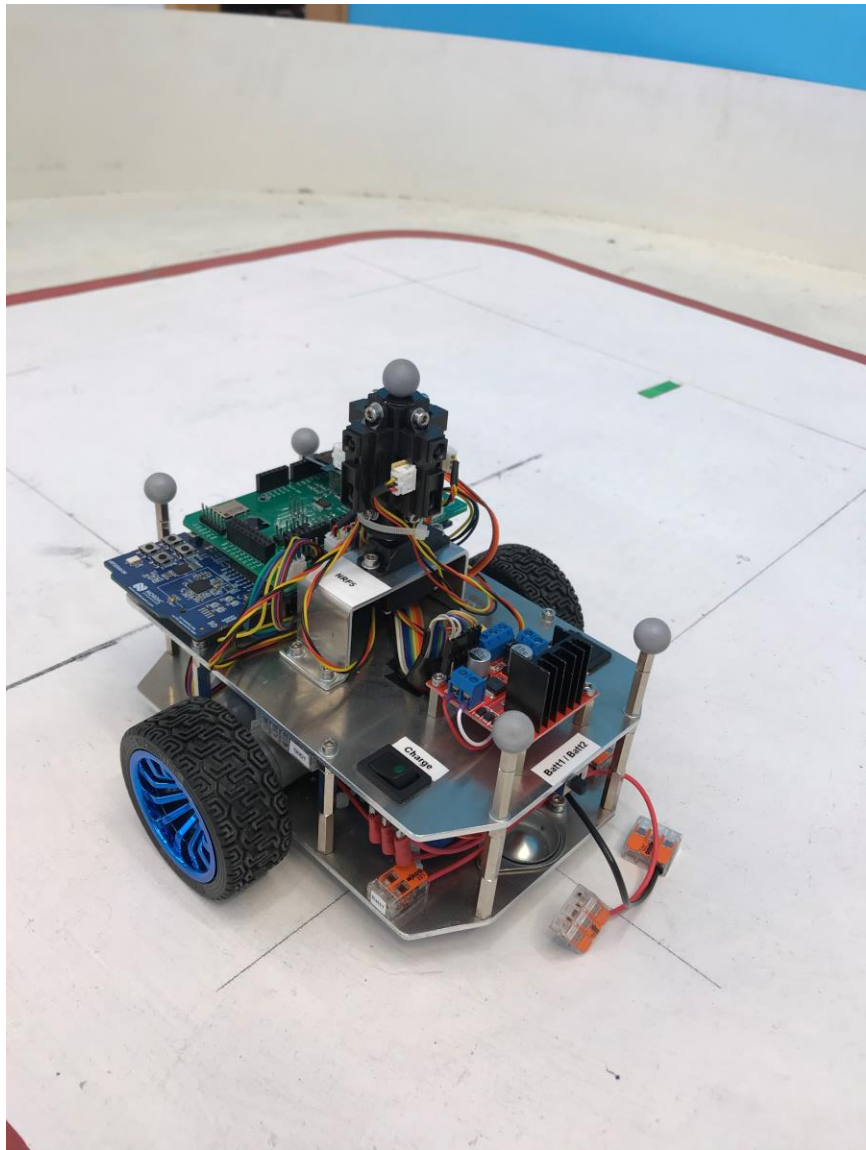
Introduksjon til «robot-prosjektet»

- Mål: Flere autonome roboter og en sentral server skal sammarbeide om å kartlegge et ukjent område



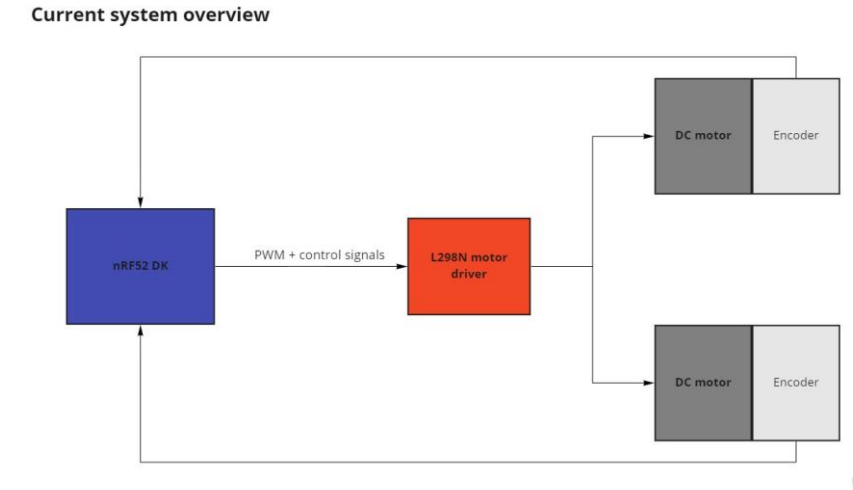
Introduksjon til «robot-prosjektet»

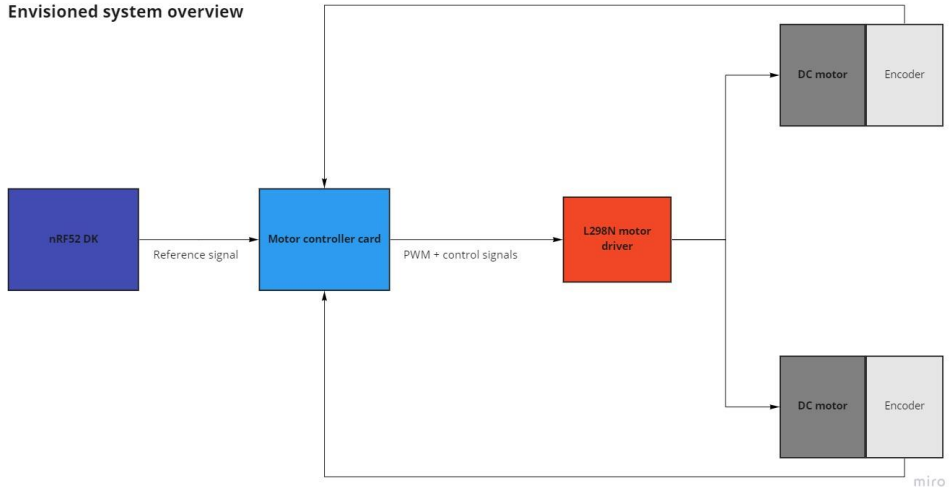
- Robot basert på nRF52840 DK
- DC motorer med enkodere
- L298N motor driver brukes som grensesnitt til motorene
  - PWM
  - Rotasjonsretning
- IR-sensor tårn
- IMU



Mål for TTK8

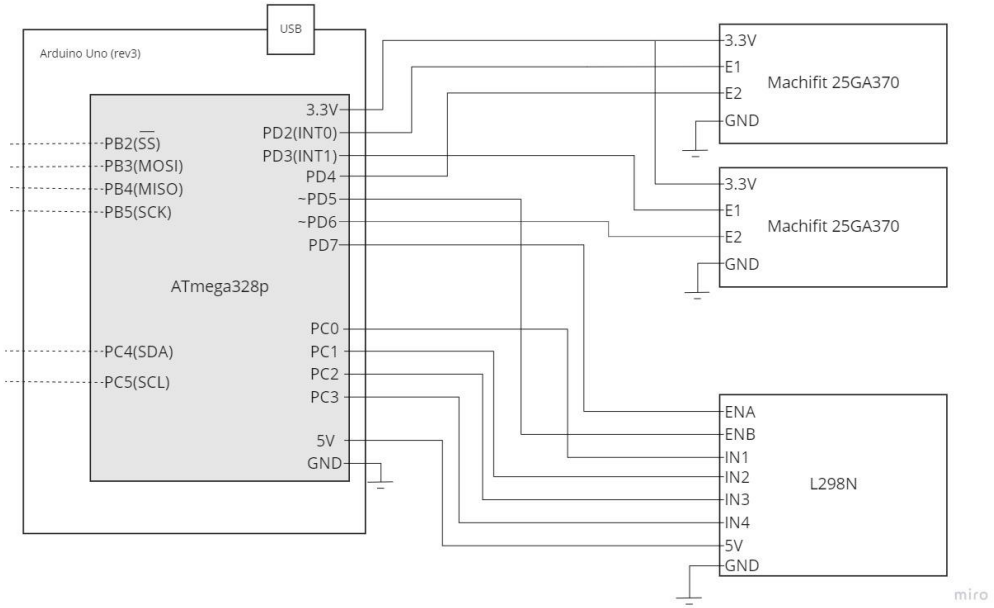
- Lage et eget kort for motor styring som skal brukes som grensesnitt for nRF52840 til motorene.
- Det nye kortet bør:
  - Ha et klart grensesnitt til roboten. Roboten skal kunne sette hastighetsreferanser og andre relevante parametre, lese hastighet fra kortet.
  - **Lese enkoderne, estimere hastighet, cruise control.**
  - Kompakt enkapsulering på PCB som skal være enkel å montere på roboten
- Formål:
  - Redusere programvarekompleksitet på nRF52840
  - Øke robusthet

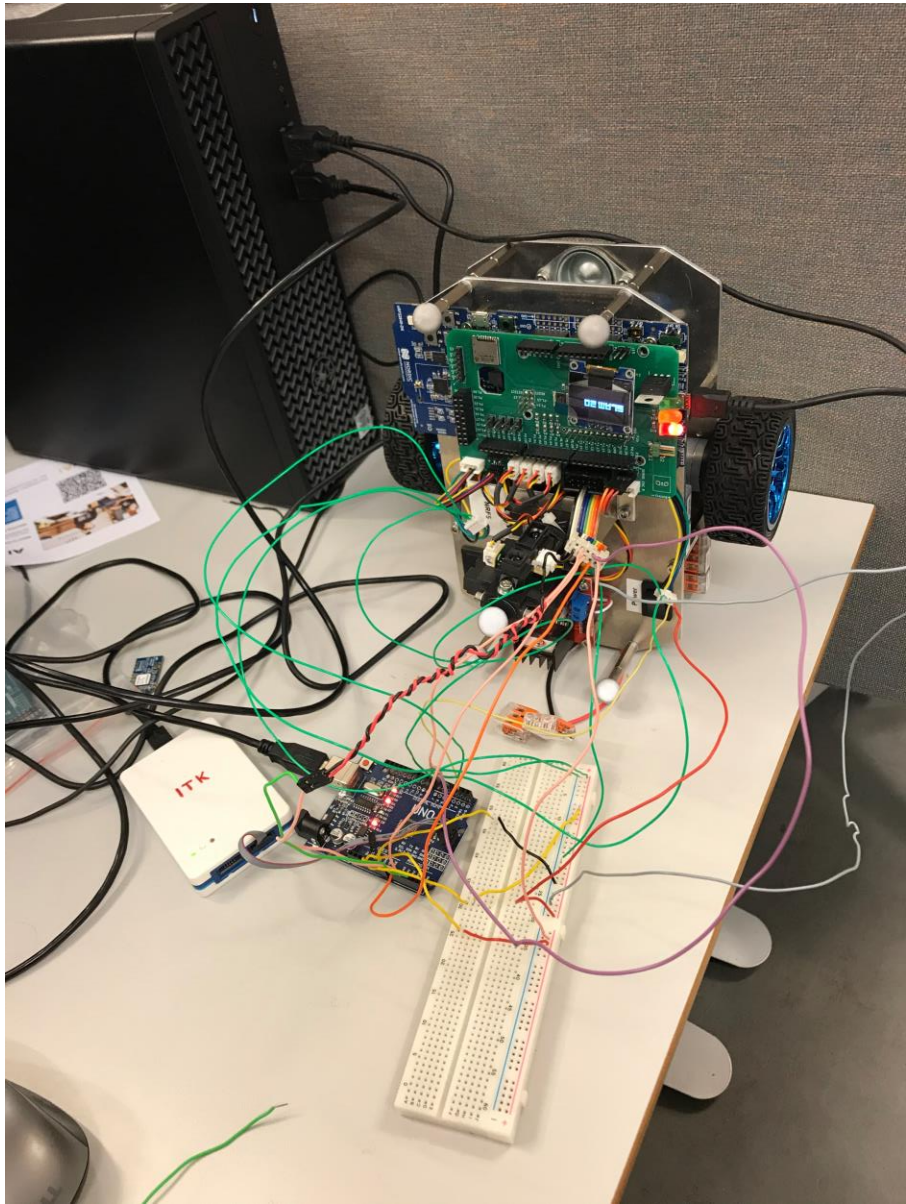




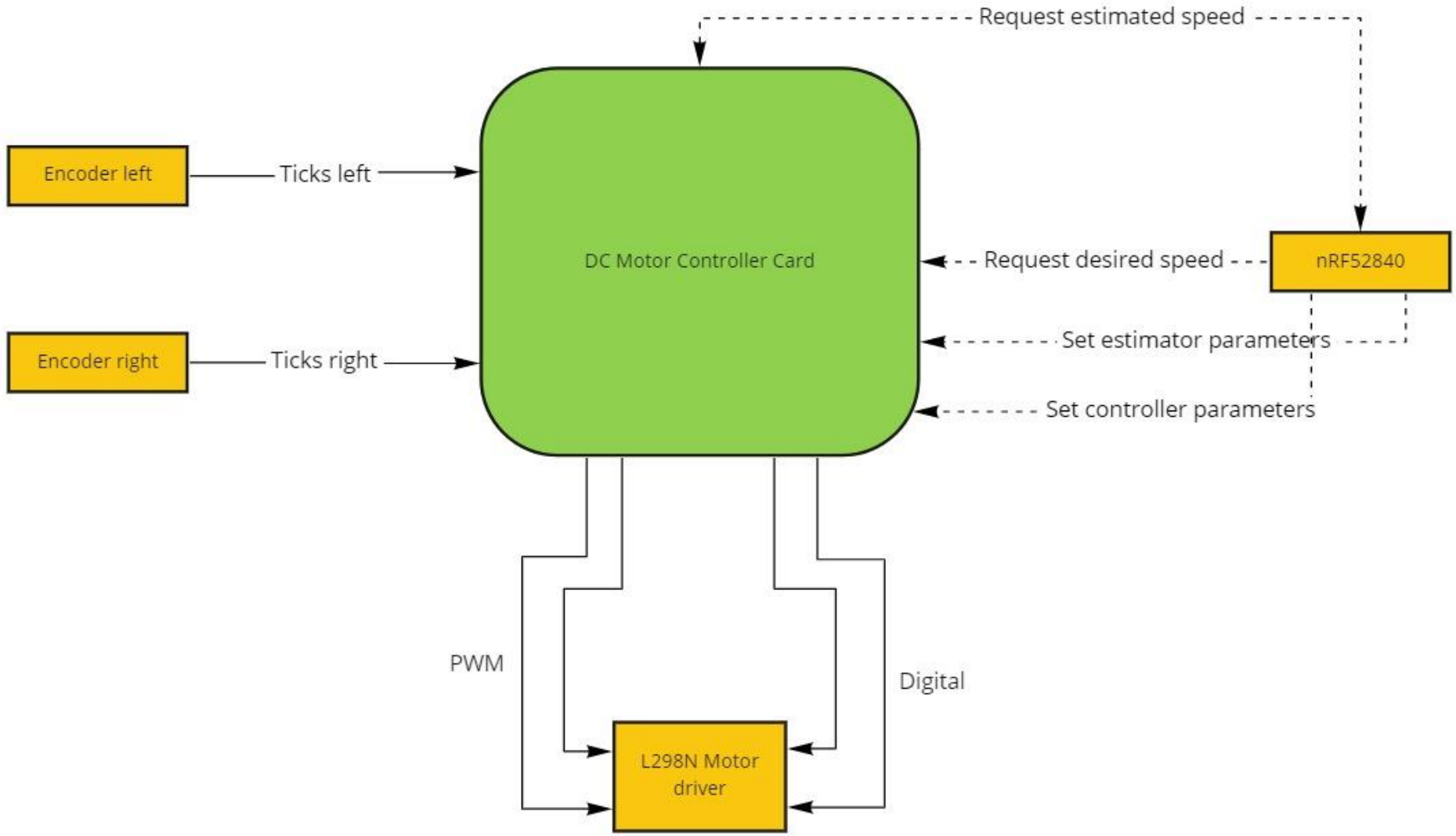
Utvikling

- Arduino Uno (rev3) for prototyping
  - ATmega328p
  - Bear-metal
- Atmel Studio + atmel ICE

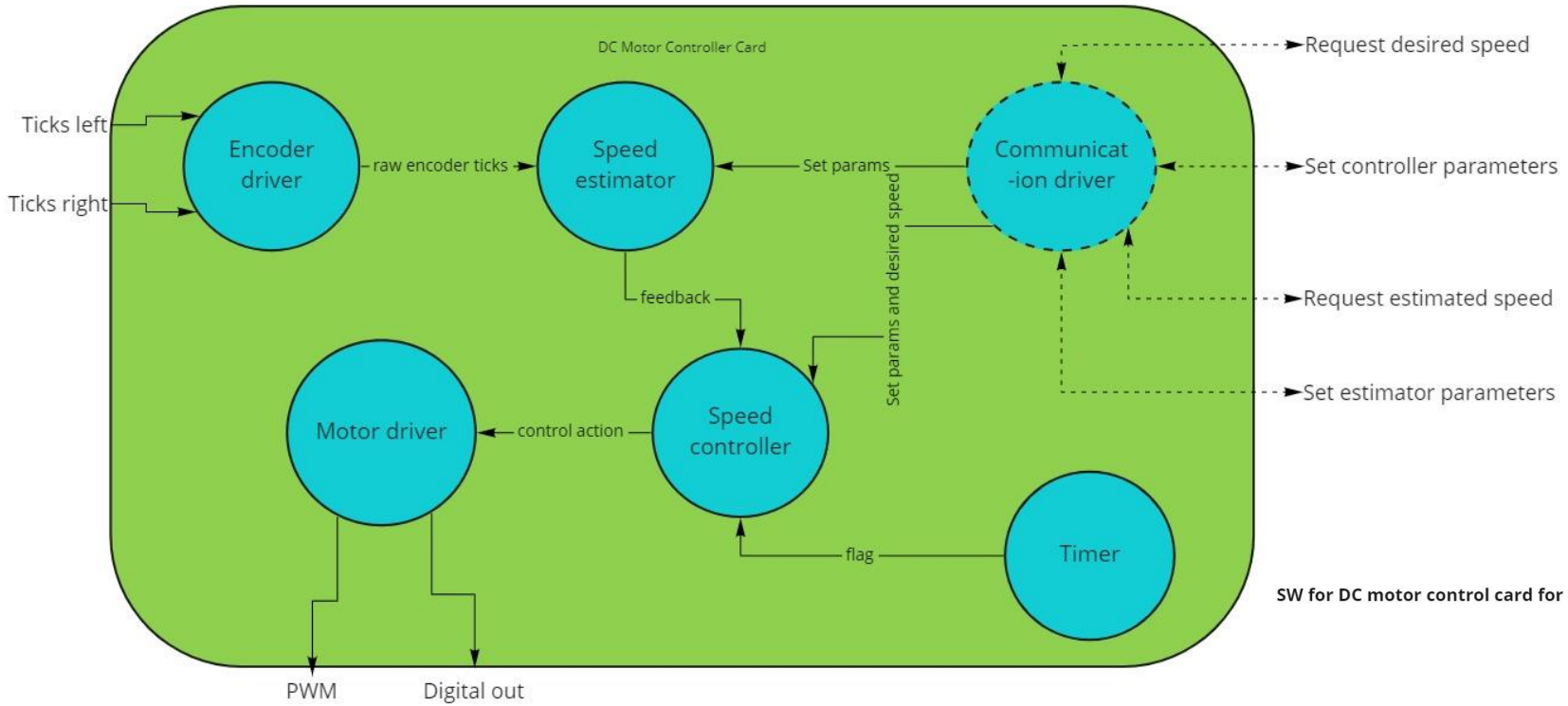




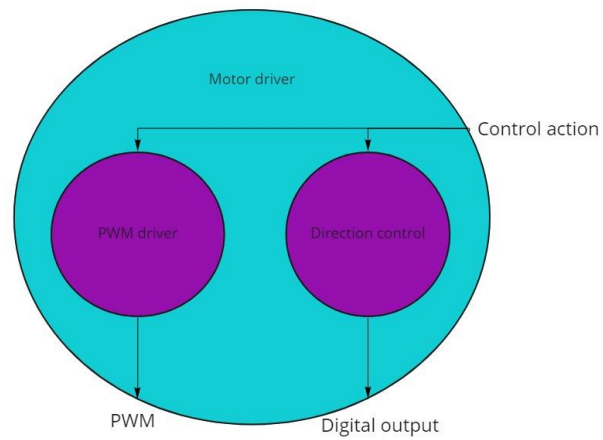
SW for DC motor control card for robot: Context diagram



SW for DC motor control card for robot: Context diagram - inner analysis 1

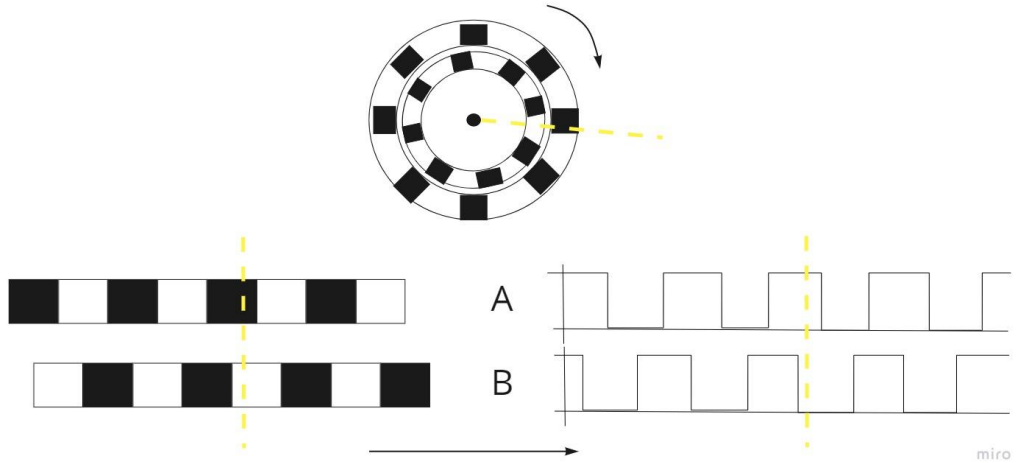


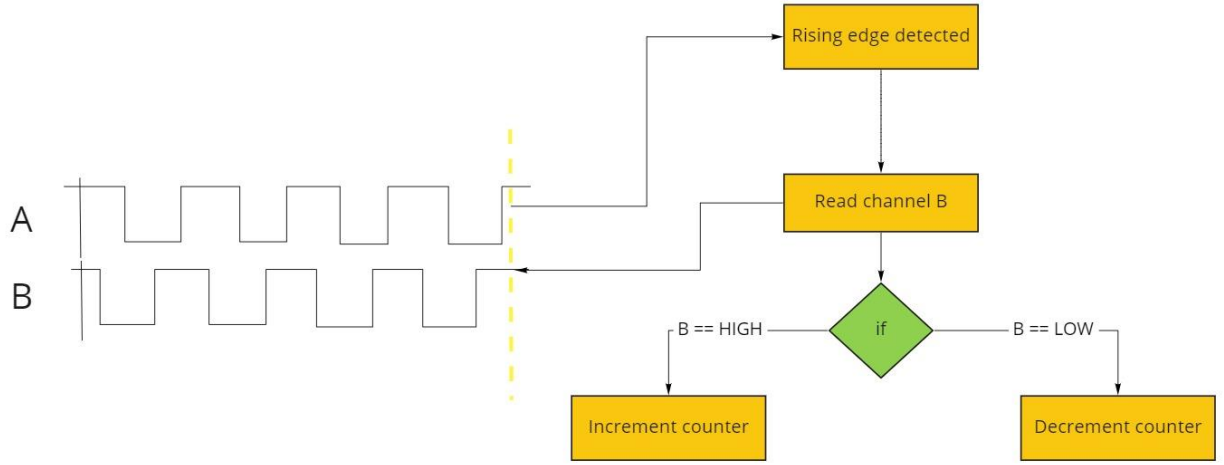
SW for DC motor control card for robot: Context diagram - inner analysis 2



Enkoder driver

- Måler hjulrotasjon
- Bruker ekstern interrupt 0 og 1
  - ISR leser lav/høy puls på en av kanalene
  - Inkrementer/dekrementer en variabel for å akkumulere enkoderpulser
- ~850 pulser / rotasjon – NB: må måles





A Encoder driver

Initialize the encoder driver.

```
void encoder_init(void);
```

Read the value of an input pin. `encoder_pin` is the desired pin to read. This function is used internally in `encoder_get_accumulated_ticks_left()` and `encoder_get_accumulated_ticks_right()`. Returns 1 (HIGH), or 0 (LOW).

```
int encoder_read_tick(int encoder_pin);
```

Returns the accumulated number of ticks from the left wheel encoder. Forward rotation of the wheel will increment the accumulated number of ticks. Backward rotation of the wheel will decrement the accumulated number of ticks.

```
long encoder_get_accumulated_ticks_left(void);
```

Returns the accumulated number of ticks from the right wheel encoder. Forward rotation of the wheel will increment the accumulated number of ticks. Backward rotation of the wheel will decrement the accumulated number of ticks.

```
long encoder_get_accumulated_ticks_right(void);
```

PWM driver

- 8-bit timer/counter0, «phase-correct mode»
- Frekvens 31 kHz
  - Må være høy nok for at L298N motor driver kortet skal kunne «glatte ut» signalet til en konstant spenning (0-12V)
  - <20kHz lager irriterende lyd
  - Bør ikke være høyere enn nødvendig → høy frekvent svitsjing i H-broen på L298N

B PWM driver

The functions defined for the PWM driver are only used internally in the motor driver module.

Initialize PWM driver.

```
void pwm_init(void);
```

Set a PWM signal to drive the left wheel. `duty` is a float representing the duty cycle of the PWM signal (0-100).

```
void pwm_set_duty_cycle_left(float duty);
```

Set a PWM signal to drive the right wheel. `duty` is a float representing the duty cycle of the PWM signal (0-100).

```
void pwm_set_duty_cycle_right(float duty);
```

USART driver

- Seriell kommunikasjon med arbeidsstasjonen
- Brukt for enkel debugging + plotting av resultater (med tilhørende python script)
- Baudrate 76800bits/s

Timer

- 8-bit timer/counter 2, CTC
- 1ms oppløsning
- Brukt for å måle tiden for andre funksjoner, og til å sørge for at hastighetskontrolleren kjører på riktig frekvens.

D Timer driver

Initialize general purpose timer. `timeout_ms` is the number of milliseconds before the timer times out.

```
void timer_init(unsigned int timeout_ms);
```

Check if the timer has timed out. Returns 1(true) if the timer has timed out, else 0(false).

```
unsigned int timer_timeout(void);
```

Reset after the timer timed out.

```
void timer_reset(void);
```

Returns time in milliseconds since `timer_init` was called. Resetting after the timer timed out will not reset the value returned from this function.

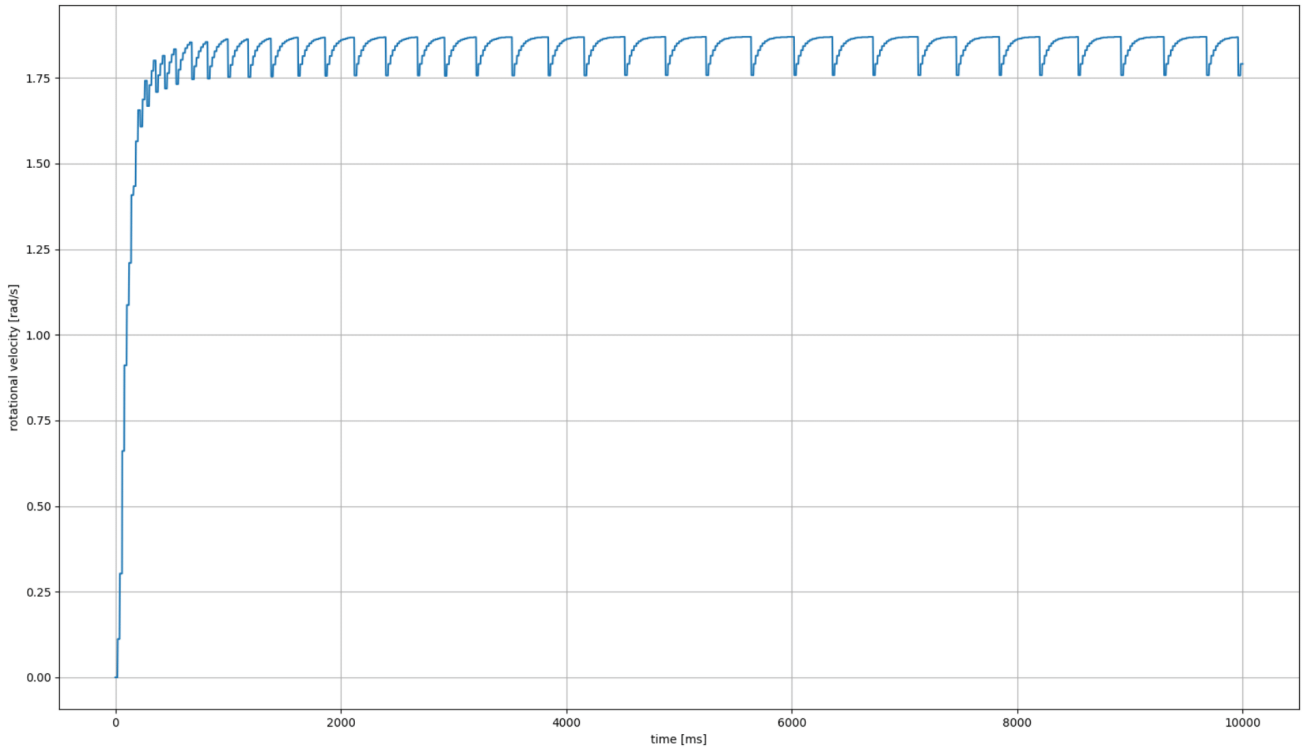
```
unsigned long timer_get_elapsed_ms(void);
```

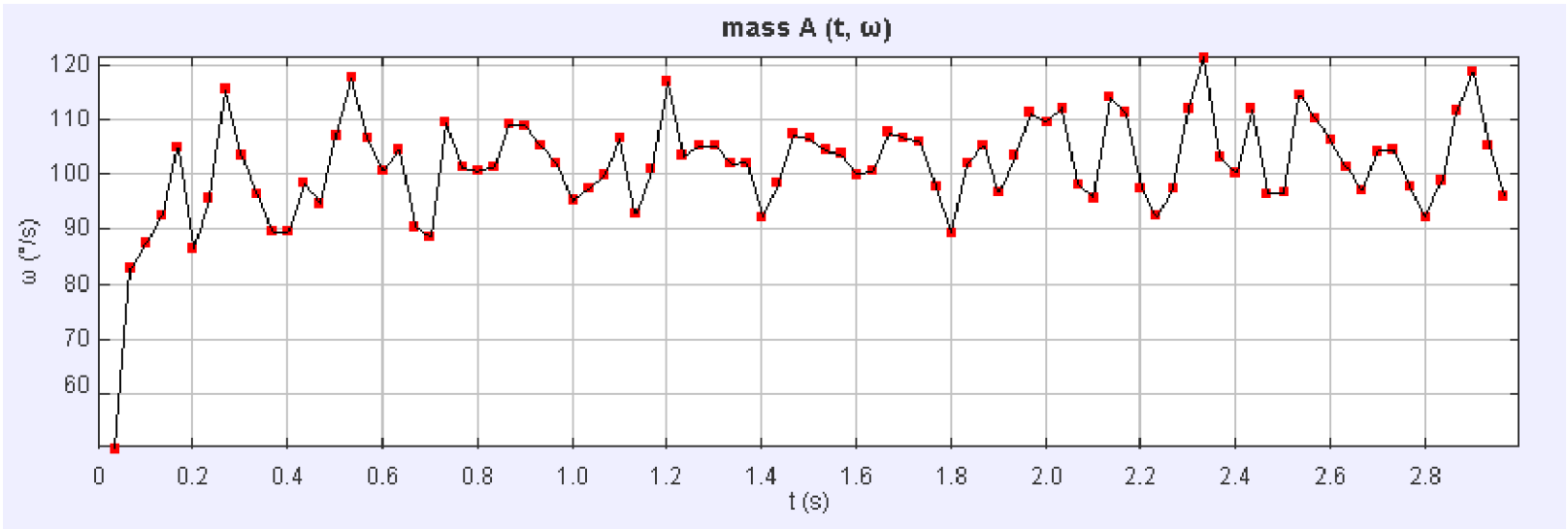
Estimering av hastighet

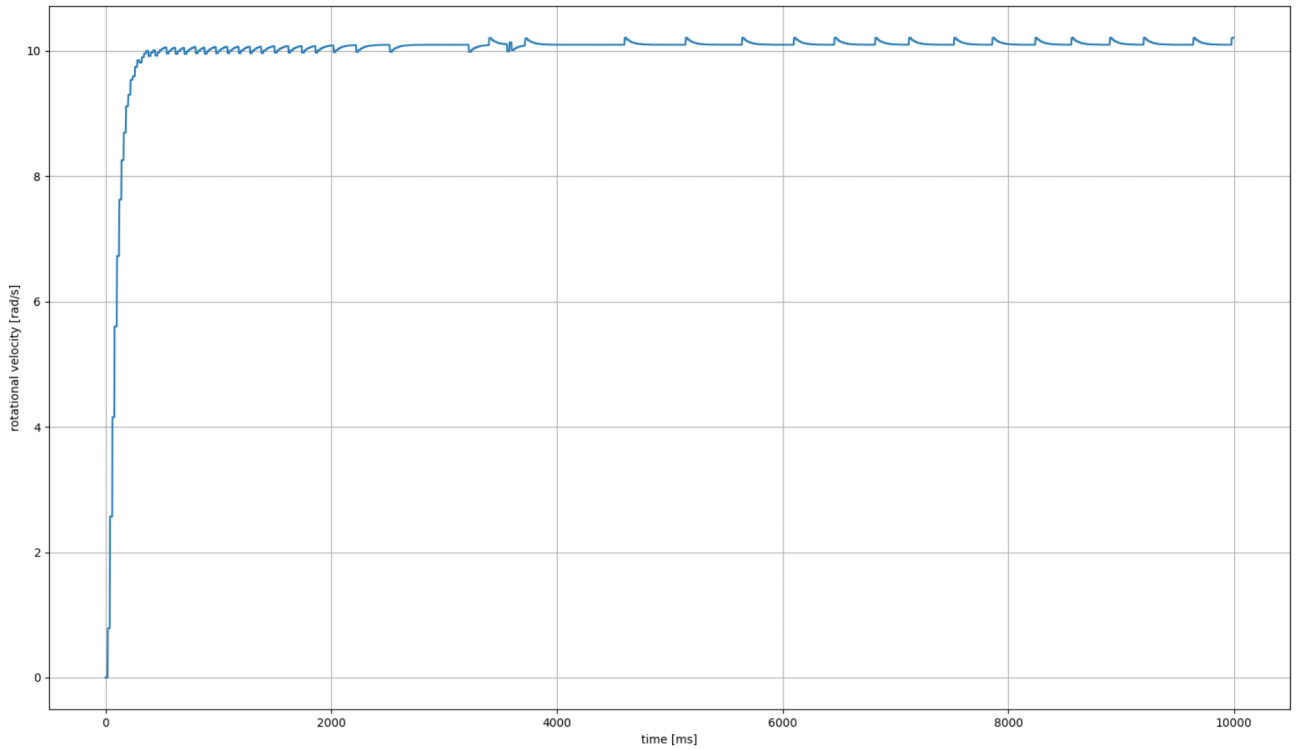
- Bruker 16-bit timer/counter 1
- CTC-mode

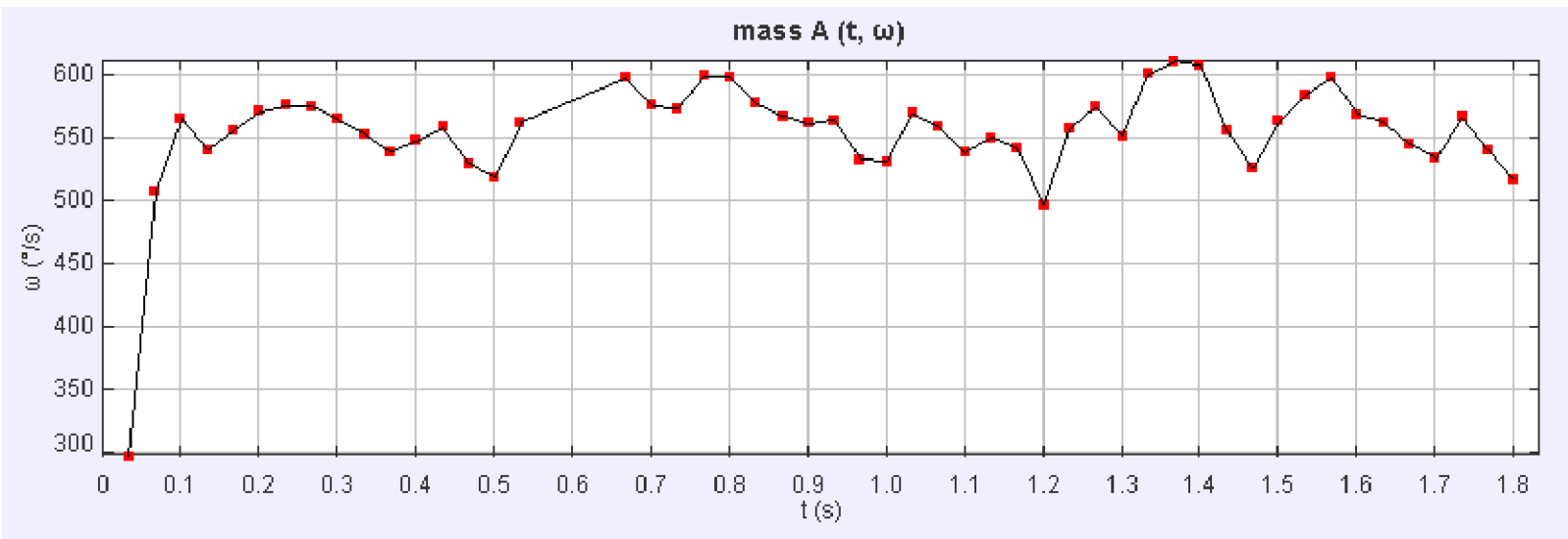
$$TargetTimerCount = \frac{TF_{CPU}}{prescalar} - 1$$

- ISR frekvens 50Hz (obs: skrevet forskjellige i rapporten)
  - Halvparten av frekvensen til enkoder pulser ved laveste hastighet
  - Sørger for at vi får minst to målinger av akkumulerte enkoder pulser
  - Bruker forskjellen i akkumulerte enkoder pulser fra forrige tidsskritt til neste.
- Digitalt lavpassfilter
  - Ved konstant hastighet måles det ikke konstant antall enkoder pulser


rotational velocity [rad/s]
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
2000
4000
6000
8000
10000
time [ms]


mass A (t, ω)
ω (°/s)
120
110
100
90
80
70
60
0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
2.0
2.2
2.4
2.6
2.8
t (s)


rotational velocity [rad/s]
time [ms]
0
2
4
6
8
10
0
2000
4000
6000
8000
10000


mass A (t, ω)
ω (°/s)
t (s)
300
350
400
450
500
550
600
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8

E Speed estimator

Initialize speed estimator. `ticks_per_rot_left` and `ticks_per_rot_right` should be the number of ticks measured by manually turning the appropriate wheel a full rotation.

```
void speed_estimator_init(long ticks_per_rot_left, long ticks_per_rot_right);
```

Returns estimated rotational velocity of left wheel.

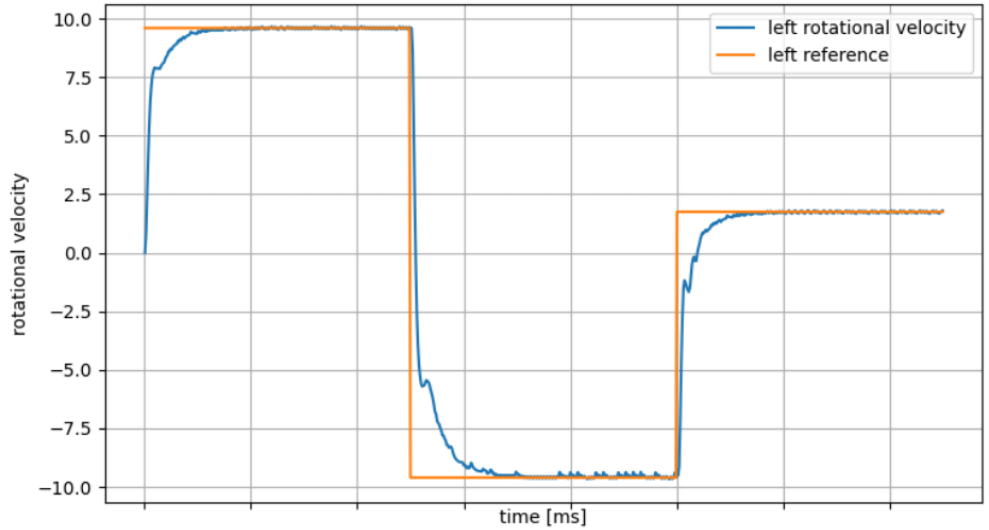
```
float speed_estimator_left_rad_per_s();
```

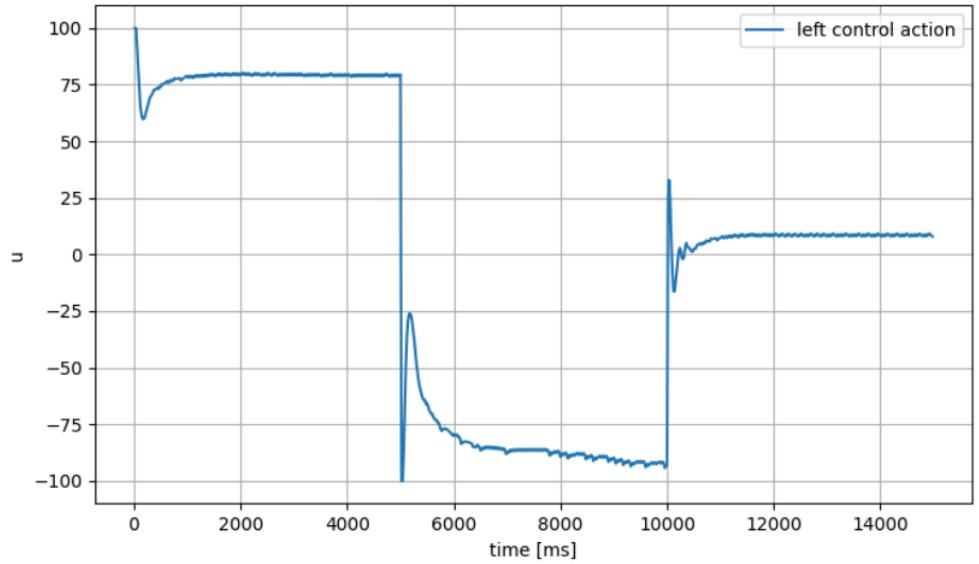
Returns estimated rotational velocity of right wheel.

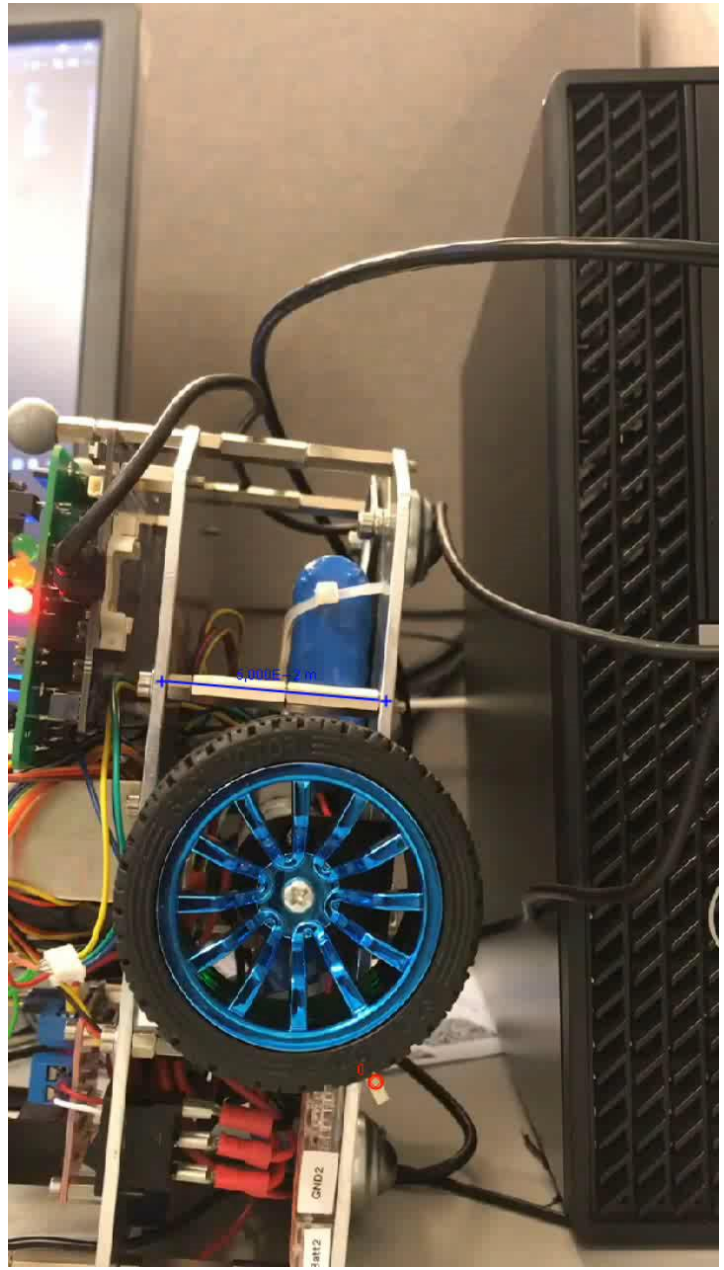
```
float speed_estimator_right_rad_per_s();
```

PID cruise control

- Styrer hastighet på hvert hjul
- Tastefrekvens 50Hz
- Vil bli brukt som indre sløyfe til å styre posisjon med nRF52840
- Anti-windup (Beard & McLain)
  - Integralfeilen akkumulerer unødvendig når motorene er i metning
  - Sørger for at når motorene er i metning holder vi integralfeilen tilstrekkelig tilbake for å holde oss på metningsgrensen.


left rotational velocity
left reference
rotational velocity
time [ms]
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0


left control action
u
time [ms]
100
75
50
25
0
−25
−50
−75
−100
0
2000
4000
6000
8000
10000
12000
14000



F PID controller

Structure for holding PID controller parameters, and internal variables.

```
typedef struct PID_controller;
```

Set controller parameters. `*pid` is a pointer to a `PID_controller`, `P` is proportional gain, `I` is integral gain, `D` is derivative gain, `loop_period` is the period of the control loop.

```
void PID_controller_set_parameters(PID_controller *pid,
float P, float I, float D, float loop_period);
```

Returns a control action to be fed as input to the motor driver.

```
float PID_controller_get_control_action(PID_controller *pid, float error);
```

Kildekode

- Tilgjengelig her: https://github.com/andersenthomas98/TTK8

Videre arbeid

- Arbeidet utført av alle studentene må samles sammen (kommunikasjon med nRF52840 og design av PCB)
- Mulighet til å tune regulator, og sette andre parametere direkte på kortet uten at dette må gå gjennom nRF52840, dvs. legg til en skjerm som kan fungere som meny + knapper for å justere innstillinger
- Posisjonsstyring fra eksternt kort istedenfor hastighet