**Dokumentation Reglerteknik**

Av Kevin, Jesper och Samuel i 180S

# Vem har gjort vad?

Jesper och Samuel samarbetade på sin arduinokod medan Kevin gjorde en egen. Kevin gjorde i ordning MQTT sidan för att bli kompatibel med bilarna och gjorde lite grundläggande styling. Jesper fixade också med stylen till sidan och implementerade även graferna.

# Jesper och Samuels styrschema

A screenshot of a cell phone

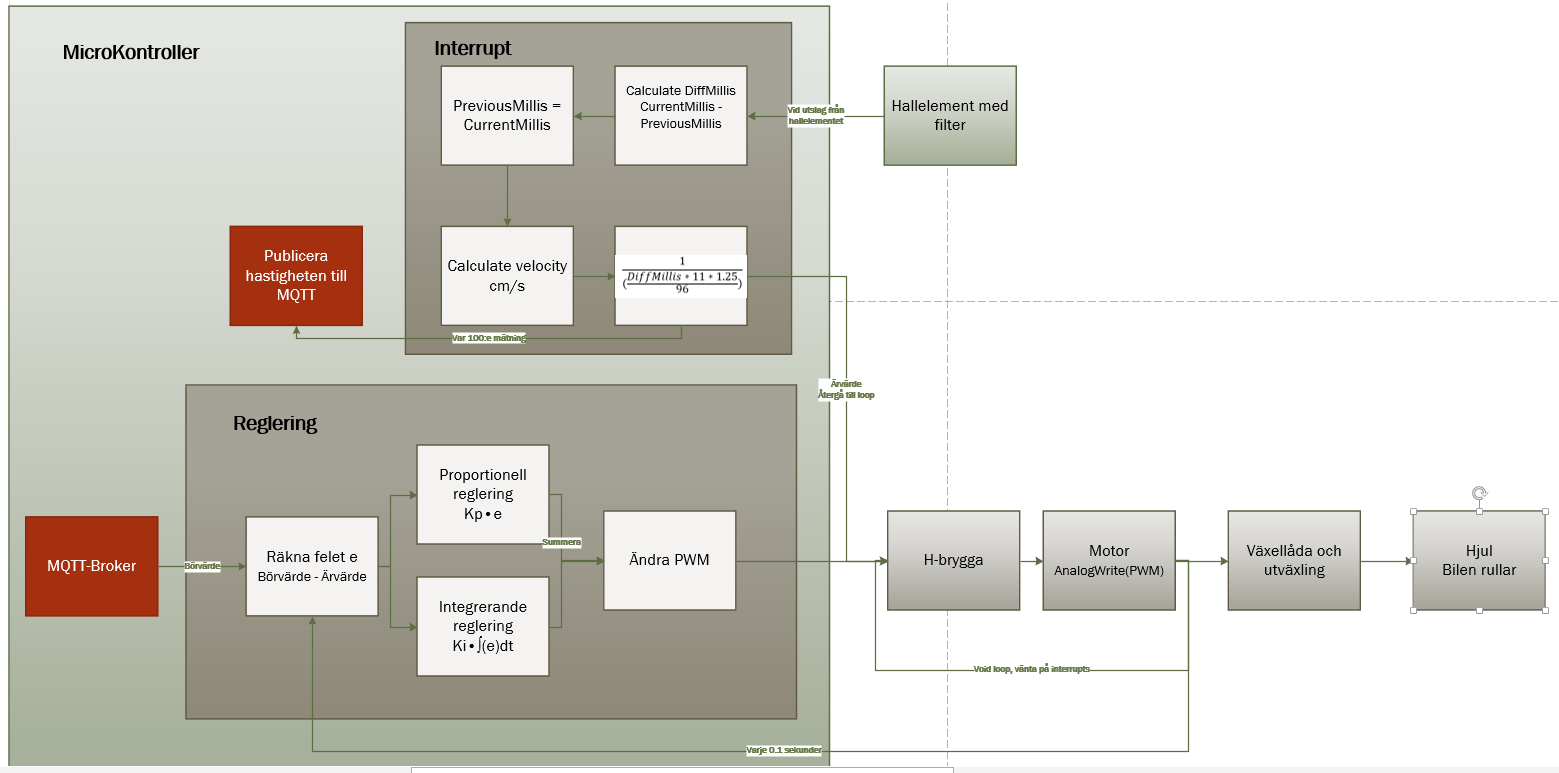
Description automatically generated

Vårt system består i huvudsak av tre delar. En mikrokontroller med motorkort, en motor och en hallgivare med filter. På mikrokontrollen körs ett program som i huvudsak utför tre uppgifter:

* Att reglera pwm-signalen ut till H-bryggan och motorn.
* Lyssna efter meddelande om nytt börvärde från en mqtt-broker.
* Räkna pulser från hallgivaren

Programmet använder sig av interrupts som kommer att lägga till en ny puls varje gång som en hög signal ändras till en låg signal från hallgivaren oberoende av var i koden den befinner sig. Under tiden som hastigheten beräknas kommer dock inga interrupts att göras, detta för att ingen tid ska hinna passera under tiden som den funktionen körs. Varje tidsenhet, i detta fall 10 ggr/sekund kommer programmet att ta antalet pulser och tiden sedan förra mätningen och beräkna rpm. Utifrån det kan även hastighet beräknas med hjälp av utväxling och hjuldiameter. Ett ärvärde skickas till reglerfunktionen som jämför det med börvärdet. Pwm-värdet ändras utifrån en proportionell och integrerande term. Mqtt-koden lyssnar efter att ett nytt börvärde skickas. I sådana fall kommer regleringsfunktionen att ändra robotens hastighet efter det nya börvärdet.

# Kevins Styrsystem



Mitt system bygger på en att en mikrokontroller (NodeMCU ESP8266) får in ett bör-värde från MQTT, som sedan jämförs med är-värdet, d.v.s. den nuvarande hastigheten, för att reglera elmotorns hastighet. Microkontrollern är ansluten till ett motorkort för att kunna styra motorn, eftersom att motorn använder en högre spänning (12V) än mikrokontrollern (7.2V i detta fall). För att få fram ett är-värde används ett hallelement med ett filter på som rensar bort störningar som är lött på ett kretskort som jag själv designat.

Sättet som hastigheten tas fram på fungerar som så att när hallelementet ändrar sin signal från 1 till 0 så kollar programmet vid vilken tidpunkt man befinner sig nu och när senaste gången man fick ett mätvärde var, och tar fram ett Δt genom att jämföra den förra tidpunkten med den nya. Sedan används lite matematik där man räknar med utväxling och hjulomkrets för att få fram bilens nuvarande hastighet. Var hundrade mätning skickas ett genomsnitt av hastigheten upp till hemsidan via MQTT.

Fördelen med att lösa hastighetsmätandet på detta vis är att hastigheten som visas alltid är den aktuella. Nackdelen är att den kan bli lite mycket för processorn att hantera, och därmed sakta ned programmet, vilket har lett till att koden behövde göras på ett visst sätt. Ett problem jag mötte var att jag inte kunde skicka upp värden alltför ofta, då det orsakade krascher på grund av överbelastning. Problemet blir då att min mätdata inte skickades lika ofta som Jespers och Samuels, vilket fick våra grafer att se lite annorlunda ut. Visserligen är det inget problem i sig, man får fortfarande fungerande grafer som går att avläsa, men det hade varit snyggare om allas grafer hade följt samma struktur. Det andra problemet som uppstår är att eftersom tiden mellan uppskickning av data beror på hur många mätningar som gjorts så kommer data skickas med olika mellanrum. Detta orsakar också problem med graferna, eftersom det inte finns någon tidsenhet att utgå ifrån. Så en möjlig förbättring till mitt program är att ändra hastighetsmätningen så att den istället skulle vara antingen tidsbunden eller att den bara gör alla uträkningar efter ett visst antal mätningar istället för varje gång. Den precision som mitt program ger är helt enkelt inte nödvändig.

Själva regleringen är tidsbaserad och sker varje 100 millisekunder, eller 10 gånger i sekunden. Här jämförs är-värdet med bör-värdet man senast fick från MQTT för att få fram ett fel. Sedan används en integrerande term och en proportionell term med detta fel. Integralen använder sig av felet på y-axeln och Δt som alltid kommer vara 100 millisekunder. Resultaten summeras och summan sätts till PWM-variabeln, som i sin tur styr hur snabbt motorn ska åka.

# A car parked in a parking lot Description automatically generatedHemsidan

Hemsidan använder sig utav en MQTT- broker hostad på maqiatto för att kommunicera med bilarna. Bilarna skickar in värden till brokern som hemsidan sedan hämtar och visar. När man trycker på ”Send Speed” skickar man in ett värde till brokern som bilarna hämtar och sätter sin hastighet efter. På bild 1 ser ni statistiksidan, där man ser bilarnas nuvarande hastighet och även graferna.

A car parked in a parking lot

Description automatically generated

Bild 2 används enbart för att ansluta till brokern. Det är enbart knapparna på bilden som är nödvändiga, de övriga textrutorna uppfyller inget syfte i nuläget.

A car parked in a parking lot

Description automatically generatedJesper har även implementerat grafer, som gör det möjligt att se hur bilarna har åkt. Detta är bra att ha om man vill analysera bilarnas resor och hur de skiljer sig åt.