Stoommachine-Aircontroller

ontwerpdocument, versie 20150609

Probleemstelling

Doel

Aansturen van de stoommachine op luchtdruk.

Oplossing om luchtkleppen te schakelen als functie van

- de slagpositie van de zuiger

met daarbij als parameters

- gewenst toerental
- toevoer luchtdruk (optioneel)

De nodige functionaliteit omvat

- noodstop
- verhinder gevaarlijke / onjuiste bediening
- slechts één cilinder aansturen
- huidige status weergeven (display en/of lampjes)
- instellingen wijzigen via menu
- startprocedure voor initialisatie, vereist wachtwoord / pincode

De oplossing moet technische opties open houden richting de toekomst.

Risicos

Op voorhand zijn deze situaties al aan te merken als aandachtsgebied voor de oplossing:

- foutieve uitlezing slagpositie / draairichting
- oversturing (te snel)
- overbelasting (te hard afremmen)
- verlies van controle luchtkleppen (stroomuitval)
- verlies van controle luchtkleppen (wegvallen luchtdruk)
- verlies van controle luchtkleppen (defecte klep, of lekkage)

Oplossing

Open-source

Dit ontwerp is “open-source”, in de zin dat het anderen aanmoedigt hierin mee te denken, of zelfs te kopieren om een eigen variant te bouwen. Mede om deze reden wordt zoveel mogelijk gebruik gemaakt van standaard oplossingen, vaak ook opgedaan uit andere “opensource” ontwerpen.
Het ontwerp (documenten, software broncode, hardware design) is toegankelijk voor iedereen:

- https://github.com/barrystaes/Stoommachine-Aircontrol

Afwegingen

De keuze valt op een microcontroller (MCU): deze kan voorspelbaar (met hoge bedrijfszekerheid) metingen en berekeningen doen. Uit tests is gebleken dat een 16MHz AVR (standaard Arduino) afdoende is voor de meting en aansturing. Echter het display en bediening mogen vriendelijker, en omdat de kosten gelijk zijn is gekozen voor een MCU met meer IO pinnen en een snellere microprocessor (80MHz ARM).

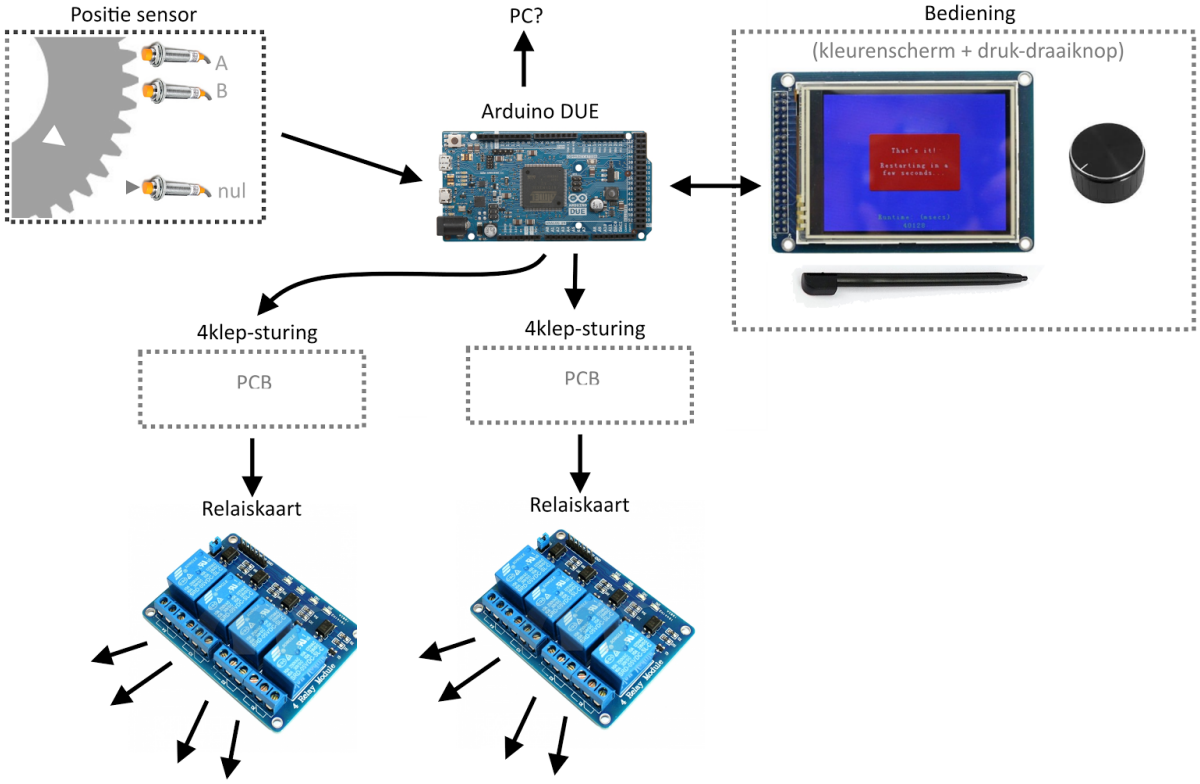
Een PLC zou gebruikt kunnen worden voor de schakeling van kleppen op basis van wiel positie, maar deze oplossing komt te kort op berekenen van parameters en dus risico beheersing.

Onderdelen

MCU	Arduino Due (3.3V)
Groot display	3.2” TFT (3.3V)
Bediening	Draaiknop voor display menu. Klepsturing 4x aan/uit/auto (per klep) Klepsturing alles aan/uit (evt. aan noodstop koppelen?) Touch (wordt vooralsnog niet gebruikt)
Klep aansturing	2x 4-Relaiskaart (5V) 2x 4-Klepsturing (3.3V naar 5V + buttons)
Behuizing	Paneelmontage in wand
Voeding	24V 150W (voor luchtkleppen en inductieve sensoren) 5V ? (voor relais) 3.3V ? (voor microcontroller)

Opstelling

Een eenvoudige weergave van de onderdelen.



Technische beschrijving

Voeding en signaal niveaus

De MCU en het display hebben 3.3V signalen, met een 5V voeding (via USB).
De sensoren hebben een 24V signaal en voeding.
De relaiskaarten hebben een 5V signaal en voeding.
Er moet dus een 24V en een 5V voeding voorzien worden.

Een sensor signaal wordt via een optocoupler op een MCU pin gezet. (24V --> 3.3V)
Een relaiskaart signaal komt via een buffer IC van een MCU pin. (5V <-- 3.3V)
Menu toetsen kunnen wellicht rechtstreeks op 3.3V worden aangesloten.

Machine positie

Om de relatieve machine positie te bepalen wordt een quadratische encoder gebruikt: inductieve sensoren A en B. Deze 2-bits “gray code” biedt een fout detectie.
Om de absolute machine positie te bepalen wordt een nulpunt gebruikt: inductieve sensor 0. Deze komt 1x per omwenteling voorbij, is vereist voor opstarten, en daarna evt. fout detectie.

De menu draai knop is ook een quadratische encoder, en wordt in software hetzelfde verwerkt.

Software taken

Om de voorspelbaarheid (lees: hogere betrouwbaarheid, kortere ontwikkel/test tijd) te bevorderen worden de MCU taken in 2 delen gesplitst:

- Sensor timer: Uitlezen van sensoren + aansturen kleppen. (“mission critical”)
- Render lus: Tekenen van live beeld.

Het uitlezen/aansturen is vastgezet op 1000/seconde. (via een timer interrupt) Dit heeft prioriteit, en onderbreekt de render lus.
In de tijd die tussendoor over is, wordt de render lus uitgevoerd. Dus het beeld getekend, seriele communicatie verwerkt, en andere langzame of onvoorspelbare dingen uitgevoerd.

Sensor timer

Door de sensoren 1000x per seconde uit te lezen wordt voorkomen dat er iets gemist wordt.
Hoe groot die kans is kan berekend worden:

De machine heeft ~168 tanden.
De 2 sensoren zien elk per tand 2 toestanden (tand / gat) dus 168 x 2 x 2 = 672 toestanden per omwenteling.
De machine gaat maximaal met 10rpm draaien, dus 10 / 60 = 0,1666 per seconde.
Dat samen maakt 672 * 0,1666 = 112 toestanden per seconde.

Dus met 1000 metingen per seconde zal elke toestand 8 keer gemeten worden. Dit geeft een hoge zekerheid, zelfs bij overspeed.

Het aansturen van de kleppen gebeurt rechtstreeks op basis van de huidige machine positie, en de gekozen openings duur. De openings duur kan afhankelijk zijn van een handmatige instelling, het gewenste RPM, het huidige RPM, de compressor luchtdruk, of (uiteindelijk) meer waarschijnlijk een combinatie van al deze factoren. Wat de beste formule is zal ook deels proefondervindelijk bepaald worden, maar we beginnen simpel.

Render lus

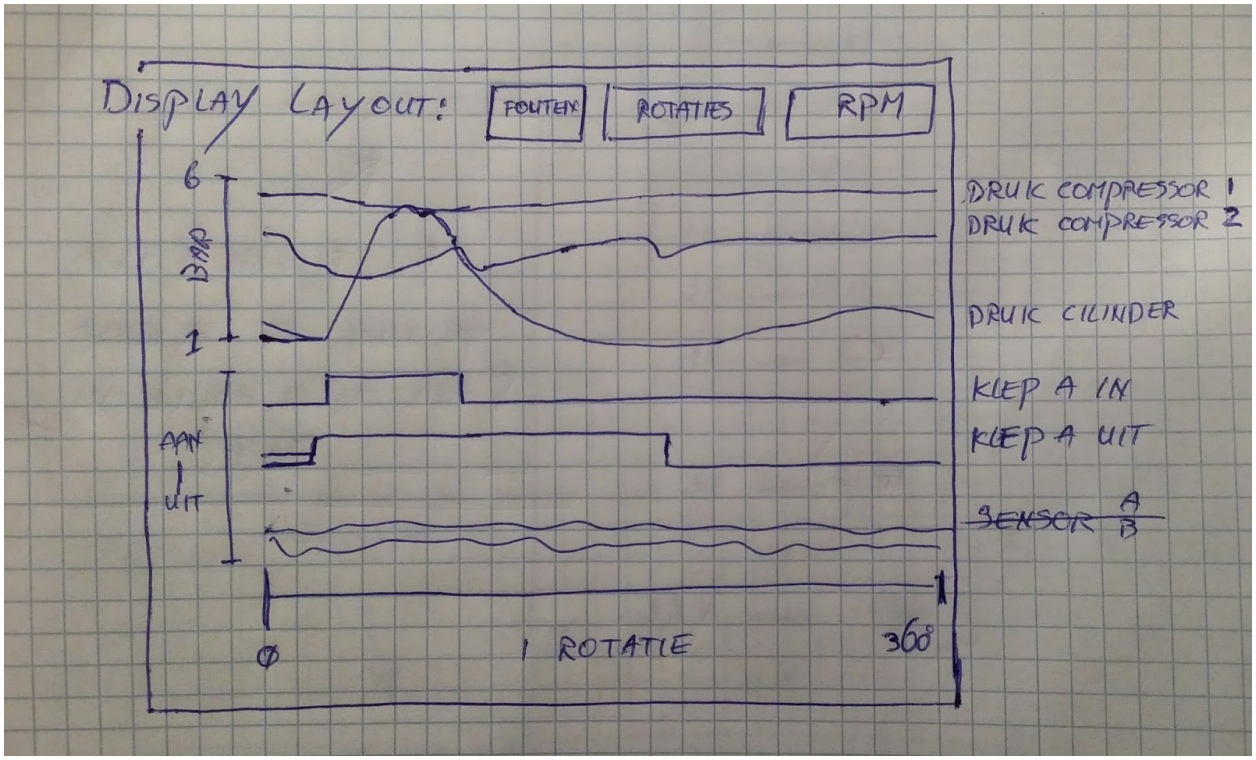
Naar verwachting kan de render lus 10 tot 60 keer per seconde volbracht worden, dus redelijk live en vloeiend beeld. Als er nog extra's bijkomen die niet missie-critiek zijn, worden die ook hier ondergebracht. Voorbeelden zijn het aansturen van display verlichting, buzzer, RGB leds, of het communiceren met een PC.

Tekenen op het display kost relatief veel tijd (in de milliseconden) dus wanneer iemand het menu gebruikt en de pagina geheel veranderd kan dit een merkbare vertraging opleveren. Dit is dan ook waarom een sensor timer gebruikt wordt.

Functionele omschrijving

Display informatie

Tijdens het gebruik wordt op het display bovenin de huidige (en gewenste) RPM slag getoond. In het midden staat een plot met daarin de instellingen en meetresultaten van de huidige slag. De plot staat dus stil, en wordt steeds overschreven. Een illustratie:



Voordat gestart kan worden moet de machine met de hand getornd worden zodat de 0-punt sensor 2x gezien is, dit is een zelf-controle.