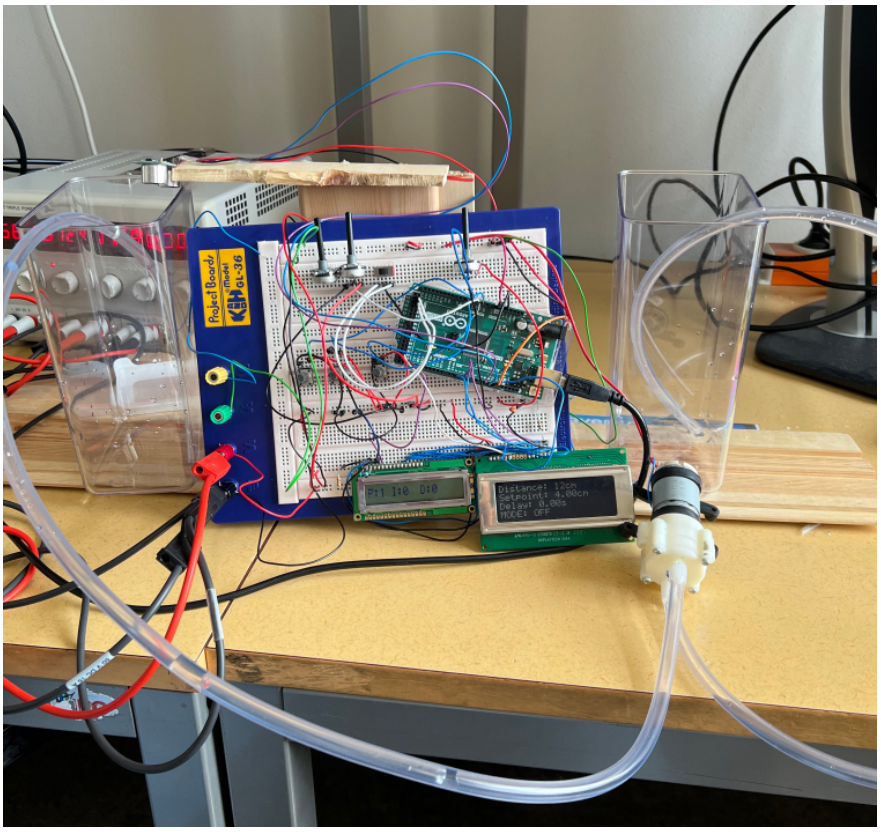
CHALMERS



Arduino-baserad vattenpumpstyrning med PID-reglering

Examensarbete inom högskoleprogrammet Elektroteknik

Kevin Abdul
Montedar Al-Ani

INSTITUTIONEN FÖR DATA- OCH INFORMATIONSTEKNIK
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg 2023
www.chalmers.se

Examensarbete 2023

Arduino-baserad vattenpumpstyrning med PID-reglering

Kevin Abdul
Montedar Al-Ani

CHALMERS

INSTITUTIONEN FÖR DATA- OCH INFORMATIONSTEKNIK
Chalmers Tekniska Högskola
Göteborgs Universitet
Göteborg 2023

Arduino-baserad vattenpumpstyrning med PID-reglering

Handledare: Lena Peterson, Data- och informationsteknik
Examinator: Jonas Duregård, Data- och informationsteknik

Examensarbete 2023
Institutionen för Data- och informationsteknik
Chalmers Tekniska Högskola
Göteborgs Universitet
SE-412 96 Göteborg
Telefon +46 31 772 1000

Omslagsbild: Systemets uppbyggnad med alla tekniska komponenter.

Skriven i LaTeX
Göteborg 2023

Arduino-baserad vattenpumpstyrning med PID-reglering
KEVIN ABDUL
MONTEDAR AL-ANI
Departement of Computer Science and Engineering
Chalmers University of Technology

Abstract

This project aims to validate the feasibility of integrating Arduino-based water pump control with PID-regulation with the help of an existing system. By utilizing Tinkercad, Arduino IDE, and various technical components, the study is conducted to provide empirical evidence that the system can indeed be successfully implemented using Arduino. Upon comparing the real system with the proof of concept system, notable similarities and differences can be observed. These differences arise from the distinct choice of components utilized in each system. The utilization of the ultrasonic sensor played a cruical role in our project, enabling us to conduct tests and obtain the achieved results. This hardware proved indispensable to our research and experimentation. Several essential actions had to be followed in order to properly build the finished product. These actions included choosing the right components, making sure that the measurements were accurate, and figuring out the system's ideal power needs. A successful outcome was made possible by each of these actions.

Keywords: arduino, water pump control, pid-regulation, tinkercad, arduino ide, proof of concept, ultrasonic sensor, hardware .

Förord

Examensarbetet som har utförts är för Högskoleingenjörsprogrammet inom Elektronik på Chalmers tekniska högskola. Projektet har utförts på Data- och informationsteknikinstitutionen. Vi vill tacka Sakib Sistek samt Veronica Olesen för projektförslaget. Vi vill även tacka Lena Peterson för att ha varit vår handledare. Dessutom vill vi också tacka Lars Norén för att hjälp oss köpa in komponenter för arbetet samt erbjudit hjälp.

Kevin Abdul och Montedar Al-Ani, Göteborg, Juni 2023

Beteckningar

Här är en lista över förkortningar som har använts i denna avhandling, listade i alfabetisk ordning:

AC	Växelström
A	Ampere
DC	Likström
D	Deriverande
GND	Ground/Jord
IDE	Integrerad Utvecklingsmiljö
IoT	Internet of Things
I	Integrerande
LCD	Liquid-Crystal-Display
P	Proportionell
PID	Proportionell–integral–derivat
PLC	Programmerbart styrsystem
V	Volt
VCC	Voltage Common Collector

Nomenklatur

Här är beteckningarna för index, mängder, parametrar och variabler som har använts i hela denna avhandling.

Variabler

u_0	Styrsignalens normalvärde
K	Regulatorns förstärkning
T_i	Integreringstiden
e	Reglerfelet
T_D	Deriveringstiden

Innehåll

Figurer

Tabeller

1
Inledning

I detta kapitel beskrivs bakgrunden samt syftet med projektet. Dessutom beskrivs kraven som är bestämda för projektet.

1.1 Bakgrund

Vattentankarna vid Chalmers tekniska högskola används för laborationer och behöver moderniseras efter över 25 års användning. För närvarande används de endast på Chalmers campus i Lindholmen. Förbättringar kan göras både när det gäller användargränssnittet och prestandan hos vattentankarna.

Vattentanken är en central komponent i kursen LEU236 Dynamiska system och reglerteknik vid Chalmers tekniska högskola. Inom kursen studeras dynamiska system som matematiska modeller för att beskriva förändringar över tid hos variabler. Reglerteknik fokuserar på att implementera styrningssystem för att reglera dynamiska processer. Vattentanken har historiskt sett fungerat som ett reglersystem och möjliggjort visuell observation av resultat i realtid. Genom att ansluta vattentankssystemet till en dator och använda externt program kan grafer och diagram genereras och visualiseras.

För närvarande används vattentankarna fortfarande med äldre teknik. Planen är att modernisera vattentankarna genom att implementera Arduino, en mikrokontroller som kommer att öka funktionaliteten.

1.2 Syfte

Syftet med projektet är att utveckla samt modernisera vattentankarna till reglertekniken med hjälp av Arduino. Panelen för vattentankarna skall också utvecklas genom att byta ut komponenter samt designen till en mer ergonomisk samt stilren design. Potentiometrarna skall också bytas ut till några som kan ge mer exakta värdesiffror. Vattentankens data skall kunna läsas av direkt med hjälp av Arduino IDE. Arduino kan också skapa grafer och sketcher.

1.3 Mål

Det mål som skall uppnås är att chalmerister skall kunna använda den moderniserade vattentanken på ett smidigt och bra sätt. Utöver det skall vattentanken bidra till att smidigare och detaljerade laborationer kan utföras på skolan. Potentiometrarna skall innehålla exakta värden som användaren enkelt skall kunna justera. Huvudmålet med projektet är att vattentankarna skall vara mer beroende av Arduino och att fler simulationer skall utföras med hjälp av Arduino. Under arbetets gång kommer paneler att ersättas och vissa komponenter kommer också att ersättas. Dessutom ska en robust panel designas med hjälp av plexiglas.

1.4 Krav/Frågeställning

Veronica, en universitetslektor vid Chalmers Tekniska Högskola, är beställaren för det genomförda projektet. Som beställare ställdes vissa krav på projektet för att det skulle anses vara lämpligt för examensarbetet. Det fanns också krav från en annan examinator som fokuserade på annan funktionalitet. Nedan visas kraven.

1.4.1 Veronicas Olesen krav

1. Koppling till pumpen (eller en annan pump) så den kan ställas in på önskat varvtal/flöde. Endast pumpen till tank 1 behöver styras. Pumpen i tank 2 är de flesta fall bortkopplad.
2. Mätning av nivå i tank 1.
3. Möjlighet att ställa in en dödtid - pumpens varvtal ändras x sekunder efter att styrsignalen ändras. Ett minimum är att dödtiden kan ställas in på 0 sekunder eller 3 sekunder. Önskvärt är även att x kan ändras i steg om 1 upp till 10.
4. Möjlighet att styra systemet (pumpens varvtal) med analog signal (0-10 V) från PLC.
5. Mätsignal motsvarande nivån i tank 1 ska kunna överföras som analog signal (0-10 V) från PLC.

1.4.2 Bertil Thomas krav (Starka önskemål)

6. Skapa en PID-regulator med inställbara parametrar. Någon typ av interface för inställning krävs.
7. Lägg in möjlighet för manuell körning av systemet. Då ska PID-regulatorn kopplas bort så den inte räknar i väg och övergång till automatisk sker störfritt. Manuell körning via en ratt (vridpotentiometer).
8. PID-regulatorn ska förses med anti-windup.

1.5 Avgränsningar

Detta examensarbete kommer inte ta hänsyn till att styra två vattentankar utan det skall begränsas till en vattentank. Andra önskade men inte nödvändiga funk-

tionaliteter såsom att plotta signaler i realtid i kurva med hjälp av ett externt matematiskt program och möjligheten att mäta kurvan för stegsvarförsök kommer inte implementeras. Detta examensarbete kommer begränsas sig till användningen av Arduinos egna program.

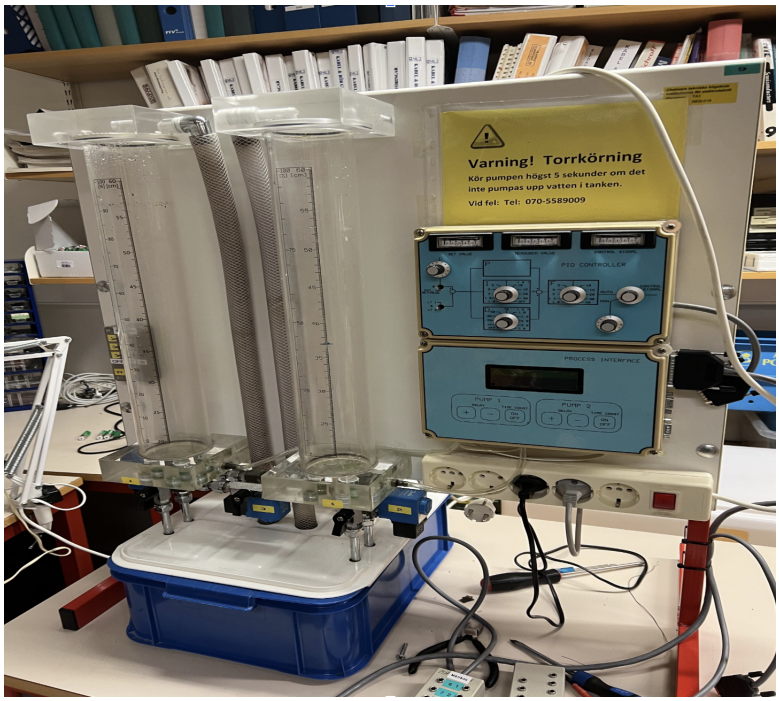
2
Teknisk bakgrund

I detta kapitel beskrivs alla tekniska komponenter som kommer att användas i projektet och deras syfte. Dessutom beskrivs de programvaror som kommer att användas under projektetsgång för att realisera systemet.

2.1 Vattentanken

På Chalmers tekniska högskola erbjuds en kurs där laborationer utförs på ett dynamiskt system, nämligen en vattentank. Kursen i fråga är LEU236 Dynamiska system med reglerteknik. Ett dynamiskt system karaktäriseras av dess förmåga att förändras över tiden och kan exempelvis vara fysiska system såsom en elektrisk krets. Inom ett dynamiskt system påverkar tidigare tillstånd och händelser det aktuella tillståndet i systemet. Reglerteknik är en ingenjörsvetenskap som ägnar sig åt att styra och reglera dynamiska system för att uppnå önskat beteende och prestanda.

Vid Chalmers tekniska högskola används ett dynamiskt system baserat på en vattentank inom ramen för kursen LEU236. Detta vattentankssystem används för regleringssyften. Den visuella representationen av vattentankssystemet återfinns i figur 2.1.



Figur 2.1: Dynamiskt system i form av en vattentank på Chalmers tekniska högskola

I den ovanstående figuren kan vi observera vattentankarna med en höjd av 60 cm. Nedanför vattentankarna finns två svarta ventiler, som kan roteras för att öppna röret och därigenom skapa en störning i fyllningsprocessen för vattentanken. Till höger om vattentankarna syns en panel där PID-parametrarna kan justeras. Under panelen med potentiometrar finns även möjlighet att ställa in systemets dödtid. Panelen på höger sida innehåller en potentiometer som möjliggör växling mellan manuell och automatisk styrning.

2.2 Funktionsprinciper

2.2.1 Vattennivåmätning

Vattennivåmätningen är en process där nivån av vattnet skall mätas. Processen kan antigen göras manuellt eller automatiskt med hjälp av en ultraljudsgivare. ultraljudsgivare är apparat som används för att mäta avståndet från ett föremål[1]. I det manuella fallet kan nivån läsas av med hjälp av en linjal som sitter fast på insidan av vattentanken. I det automatiska fallet kan nivån läsas av med hjälp av arduinon som är uppkopplad med ultraljudsgivare.

2.2.2 Programmerbart styrsystem

Programmerbara styrsystem (PLC) är en typ av datorbaserade styrningssystem som används för att automatisera processer inom industriell produktion och andra liknande tillämpningar[1]. PLC-systemen består vanligtvis av en central enhet (CPU)

som är programmerad för att övervaka och styra reläer, motorer, sensorer och andra elektriska enheter som används i produktionsprocessen. Dessa system är konstruerade för att kunna hantera olika typer av ingångar och utgångar, vilket gör dem lämpliga för många olika applikationer.

En av fördelarna med PLC-system är att de är mycket tillförlitliga och kan fungera under extrema förhållanden. Dessutom är de enkla att programmera, vilket gör dem till ett populärt val för många applikationer. PLC-system kan också enkelt integreras med andra datorsystem, vilket möjliggör mer avancerade automatiseringslösningar[1].

En annan fördel med PLC-system är att de är lätta att underhålla och uppgradera. Eftersom de är modulära kan enskilda delar enkelt bytas ut vid behov, vilket minskar kostnaderna för underhåll och reparation. Dessutom kan nya funktioner enkelt läggas till genom att uppdatera programvaran som körs på CPU:n[1].

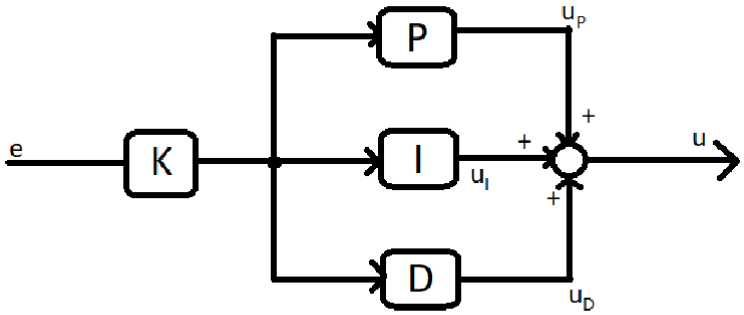
Sammanfattningsvis är programmerbara styrsystem (PLC) viktiga verktyg inom industriell produktion och automatisering[1]. De är pålitliga, enkla att programmera, lätt att underhålla och kan integreras med andra datorsystem. Med fördelarna som PLC-system erbjuder är det inte förvånande att de är så populära inom industriella applikationer.

2.2.3 Reglerteknik

För att driva ett autonomt system krävs det reglerteknik. Reglerteknik är tanken bakom hur ett autonomt system kan bedrivas med en passande metod [2]. Det automatiska systemet ska kunna agera utan mänsklig övervakning samt bidra till en smart lösningsmetod. Reglertekniken innefattar matematiska modeller som tillämpas på det angivna systemet som används för att beskriva systemet beteende. De matematiska modellerna utformar önskvärd stabilitet samt prestanda hos det angivna systemet. De flesta tillämpningar kräver att det automatiska systemet håller koll på variabler i systemet [2]. Inom reglertekniken finns det flera metoder som tillämpas för att utföra det automatiska systemet och de mest använda metoderna innehåller proportionell–integral–derivata (PID) regulator.

2.2.4 PID-regulator

I en PID-regulator består styrsignalen av tre bidrag [2]. Dessa bidrag är P-delen som är det proportionella bidraget. I-delen som är integrerande bidraget samt D-delen som är derivata bidraget. PID-regulatorn är den mest förekomna typen av regulator inom industrin [2]. För att få en enklare förståelse av PID-regulatorn så kan den ritas upp som ett blockschema. I blockschemat ingår det alla tre bidrag som behövs för att bilda PID-regulatorn.
Blockschemat i Figur 2.2 visar summan av de tre bidragen bildar en utsignal u med insignalen e. Insignalen e är systemets reglerfel. Reglerfelet är skillnaden mellan önskade samt uppmätta värdet [2]. PID-regulator använder reglerfelet för att sedan styra och justera systemet. Summan u är varierande beroende på det insatta värdet



Figur 2.2: PID-Regulator Blockschema

av P, I och D-verkan. Värdet påverkar hur regulatorn fungerar.

Den proportionella regleringen (P-reglering) förekommer hos de flesta regulatorer och är oftast kombinerad med integrerande regleringen [2]. P-reglering bidrar till att svängningar på utsignalen undviks. Vid P-reglering ingår det två konstanter u_0 samt K. Börvärdet är u_0 och K används för förstärkningen av systemet som skall regleras. K används för att rätta till fel som uppkommer vid körning och beroende på hur högt värdet på K är kan stabiliteten samt snabbheten variera. Ett lågt K värde bidrar till stabilare men segare system, och ett stort K värde bidrar till ett snabbt system som är instabilt. Nackdelen med den proportionella regleringen är att det ger kvarstående fel vid stegformade börvärdes-ändringar och därför behövs Integrerande reglering.

Det integrerande regleringen (I-reglering) är uppbyggt av integreringstiden T_I som bestämmer hastigheten på integreringen av följande ekvation $\left[\frac{1}{T_I}\int_0^t e(t)\right]$. Ekvationen motsvarar e reglerfelet samt t tiden över hur långt reglerfelet varar. I-regleringen är beroende av värdet på T_I då det är värdet på T_I som bestämmer hastigheten på regulatorn. Det integrerande regleringen används för att reducera fel vid stegformade störningar i det angivna systemet[2]. P-reglering jämfört med I-reglering är att det reagerar på fel direkt medan för I-regleringen tar det tid att reagera på fel.

Den deriverande regleringen (D-reglering) är olik de andra typerna av reglering. Det är den enda regleringsmetoden som ej kan användas ensamt utan måste kombineras med I-reglering eller P-reglering [2]. D-reglerings primära syfte är att öka stabiliteten, snabbheten samt störningsdämpningen. Detta är möjligt med Deriveringstiden T_D som är en konstant. D-blocket utsignal är ej lika med noll då derivatan på insignalen ändrar sig och desto snabbare ändring desto större utsignal. Kortvariga störningar elimineras med D-reglering dessutom motverkar D-reglering översvängar samt instabilitet. Nackdelen med D-reglering är att den är mycket mer känslig och reagerar fort på störningar dessutom krävs det kraftigare styrsignaler därför bidrar det till en känsligare regulator.

PID-reglering är extremt vanligt förekommande och det beror på att användaren

inte behöver vara bekant med avancerade matematiska beräkningar [3]. Utöver det har PID en lång historia som många ingenjörer är bekanta med. Användningen av PID har blivit en standard i industrin. Digital styrning har också förbättrat PID:s självinställning och förstärkningsplanering. PID-reglering löser många reglerproblem samt är flexibelt till att anpassas för det systemet som regleringen införs på. Regulatorn är uppbyggd på ett samband som kan skapas med figur 2.2. Blockschemat som innehåller proportionella, integrerande samt deriverande verkan och är uppbyggt på ekvationen $u(t) = K\left[e(t) + \frac{1}{T_I}\int_0^t e(t)dt + T_D * e'(t)\right]$. Med sambandets parameterar K,T_I och T_D kan regulatorn anpassas.

2.3 Justering av PID-parametrarna

Bestämmelsen av PID parametrarna kan göras manuellt via trial and error", också kallad manuell inställning. Däremot finns det ett smidigare sätt att ställa in PID parametrarna och det är via tuningmetoder. Parametrarna kan bestämmas för en specifik tillämpning. Det finns flera olika metoder att utföra justeringen men de mest populära och använda är Ziegler-Nichols samt Cohen-Coon.

2.3.1 Ziegler-Nichols

Ziegler och Nichols var två professorer som kom på ett sätt att optimera PID-regulatorer och är en så kallad stegsvars-/frekvenssvarsmetod. Metoderna har fördelen att de inte kräver tidskrävande empiriska beräkningar eller omfattande provningar. Regleringar med denna metod blir ofta instabila och oscillerande. PID parametrarna bestäms av K_P, tidsintegral konstanten T_I samt deriverings tiden T_D[4]. För att justera systemet enligt Ziegler Nichols så behövs två variabler och de är a och L som även ses i tabell 2.1 nedanför.

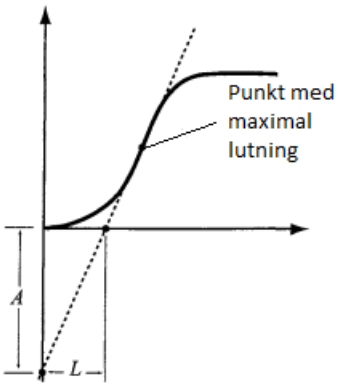
Tabell 2.1: Justerings Parameter Ziegler-Nichols

	Kp	Ki	Td
P	$\frac{1}{a}$		
PI	$\frac{0.9}{a}$	3L	
PID	$\frac{1.2}{a}$	2L	$\frac{L}{2}$

För att beräkna variablerna för Ziegler Nichols stegssvarsmetod krävs det ett stegsvars diagram för funktionen som används[4]. Ett exempel på stegsvar som förekommer med fördröjning kan visas i Figur 2.3.

2.3.2 Cohen-Coon

Cohen-Coon (CC) justeringsmetoden skapades av Cohen och Coon 1953. PID parametrarna bestäms av reaktions kurvan på den angivna funktionen [5]. CC är uppbyggd på Ziegler-Nichols justerings-metod däremot är tillämpningen mer komplex med bredare matematiska beräkningar. CC är också inte lika begränsad som Ziegler-Nichols och kan användas för fler system [6]. CC fungerar bäst på system med dödtid som är hälften av funktionens tidskonstant. Tidskonstanten är en parameter



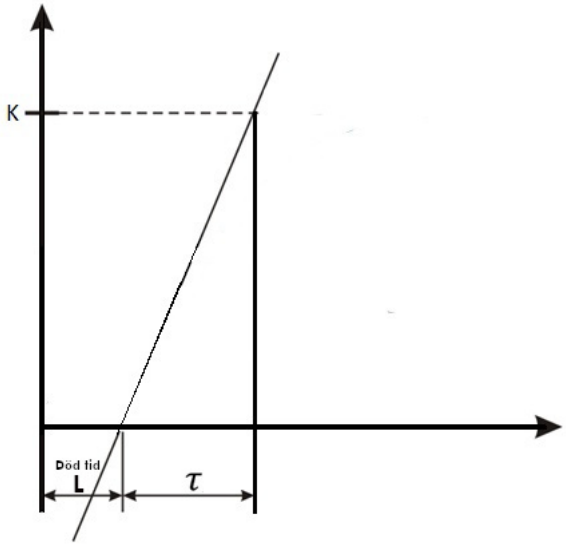
Figur 2.3: Grafisk bestämning av parametrarna A och L

som används för justerings av regulatorns stabilitet och respons. PID parametrarna bestäms av den proportionella förstärkningen K_P, tidsintegral konstanten T_I samt deriverings tiden T_D som beskrivet tidigare. För att justera ett system behövs tre variabler beräknas och de är K, tau (τ) samt L som motsvarar död-tiden. Formlerna för systemet visas i tabell 2.2.

Tabell 2.2: Justerings Parameter Cohen-Coon

	K_P	T_I	T_D
P	$\frac{1}{K} * \frac{\tau}{L}\left(1 + \frac{L}{3\tau}\right)$		
PI	$\frac{1}{K} * \frac{\tau}{L}\left(\frac{0{,}9L}{12\tau}\right)$	$\frac{L\left(30+\frac{3L}{\tau}\right)}{9+20\frac{L}{\tau}}$	
PID	$\frac{1}{K} * \frac{\tau}{L}\left(\frac{16\tau+3L}{12\tau}\right)$	$\frac{L\left(32+\frac{6L}{\tau}\right)}{13+8\frac{L}{\tau}}$	$4\frac{L}{11+2\frac{L}{\tau}}$

För att beräkna variablerna krävs ett stegsvar-diagram eller en reaktions-diagram för den angivna funktionen. De tre variablerna som krävs för beräkningen av parametrarna kan alla tas fram från stegsvar diagrammet. Ett exempel på ett stegsvar diagram med fördröjning kan visas på Figur 2.4.



Figur 2.4: Grafisk bestämning av parametrarna K, L och τ

Variabeln K är förstärkningen av det angivna systemet och i detta fall har värdet 1. τ beräknas genom att markera ut inflexionspunkten på systemet och skapa en tangent. Ett tydligt exempel på hur det kan ske är visat på Figuren 2.4. L är hur stort systemets dödstid är och är bestämt beroende på det.

2.4 Mjukvara

2.4.1 Tinkercad

Tinkercad är en mjukvara som används för att få en djupare förståelse kring de tekniska komponenterna. Utöver det används Tinkercad för att få kunskap om programmering samt hur de olika komponenterna kan arbeta med varandra och kontrolleras med programmeringsspråket C++[7].

Med hjälp av simulationer kan användaren undersöka hur komponenterna fungerar. Detta bidrar till att man inte skadar verkliga komponenter samt utför detaljerade tester [7]. Tinkercad är ett webbaserat verktyg vilket bidrar till att användaren kan få snabb tillgång till sina kreationer.

Tinkercad har en bred komponentlista som bidrar till att användaren kan vara kreativ med att skapa och testa sitt projekt till att senare realisera det. Tinkercad används som ett verktyg till att skapa en grund för sitt projekt och för att testa olika lösningar till problemen som kan uppstå. Detta sparar användaren tid med att förenkla kopplingar samt utföra snabba simuleringar.

2.4.2 Arduino-IDE

Arduinos integrerade utvecklingsmiljö, även känd som Arduino IDE, är en mjukvarumiljö som används för att skapa och ladda upp program till Arduino-kort. IDE:en är baserad på en öppen källkods plattform [8] och är utformad för att vara användarvänlig för både nybörjare och avancerade användare.

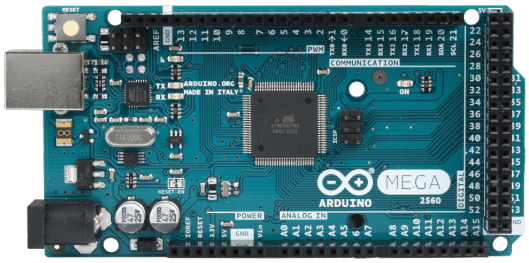
IDE:en har en textredigerare som gör det möjligt att skriva kod och en integrerad kompilator som översätter koden till maskinkod som kan köras på Arduino-kortet. IDE:en stöder också en mängd olika bibliotek och verktyg som gör det lätt att använda olika sensorer och moduler med Arduino-kortet [9].

En av fördelarna med Arduino IDE är att den är tillgänglig för flera olika plattformar, inklusive Windows, Mac och Linux [10]. Dessutom finns det också många andra alternativa utvecklingsmiljöer tillgängliga för Arduino, inklusive PlatformIO och Eclipse Arduino IDE [11].

2.5 Komponenter

2.5.1 Arduino-Mega

En Arduino Mega 2560 är en mikrokontroller som styr med hjälp av ett mikrochip av märket AVR ATmega2560 [12]. Mikrokontrollen är väl ägnad till projekt som kräver ett stort antal ingångar och utgångar som kan ses på Figuren. 2.5



Figur 2.5: Arduino-Mega 2560

Arduino har potentialen att användas i olika typer av projekt, både stora och små. En av fördelarna med Arduino är dess förmåga att hantera ett stort antal in- och utgångar. Den har till exempel 16 analoga ingångar och 54 digitala in-/utgångar, varav 15 kan styras med pulsbreddsmodulering. Denna funktionalitet gör Arduino till en flexibel plattform som kan anpassas efter behoven hos olika projekt, oavsett om det är prototyper, hobbyprojekt eller mer omfattande professionella projekt. [12].

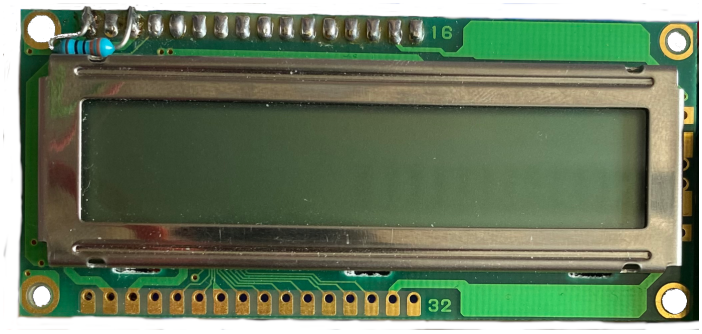
2.5.2 Potentiometer

Potentiometern är en komponent som behövs för att realisera vattentanksystemet. Potentiometern är ett mätverktyg och används som en steglös variabel resistor med spänningsdelare [13]. Det som gör potentiometern till ett unikt motstånd är att

den inte drar någon ström och har en oändlig impedans. Potentiometern har tre pinnar som är anslutna till ett resistivt element och en justerbar kontaktyta. De tre pinnarna som finns ska anslutas till jord, Spänningsgemensam kollektor (VCC) och utsignal. Potentiometern används främst inom spänningsreglering. Potentiometern kan oftast vridas i totalt 270°. Med ekvationen $\frac{grader}{270} * V$ kan spännings motståndet regleras.

2.5.3 LCD-skärm

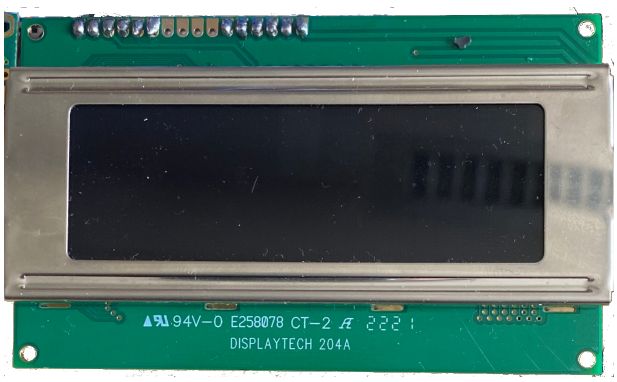
LCD-skärmen, visad nedanför i figuren, är en plattskärms eller annan elektronisk modulerad optisk komponent. LCD-skärmen använder sig av ljus modulerande egenskaper gjorda av flytande kristaller kombinerat med polarisering [14].



Figur 2.6: Liquid-Crystal skärm 16x2

De flytande kristallerna inuti skärmen är inte det som ansvarar för att emittera ljuset, i stället använder sig skärmen av svartljus eller reflektorer för att producera bild på skärmen. LCDs används oftast i syfte att visa arbiträra bilder (som i en generell bildskärm) eller fixerade bilder med låg informationsöverföring [14].

En annan vanlig modell av LCD-skärmen är en 20x4 LCD-skärm, som har kapacitet att visa upp till 20 tecken i bredd och 4 rader av tecken och kan ses i nedanför Figur 2.7. Precis som 16x2 LCD-skärmen använder 20x4 skärmen sig av flytande kristaller för att manipulera ljus genom polariserade material och visa bilder eller text[14].



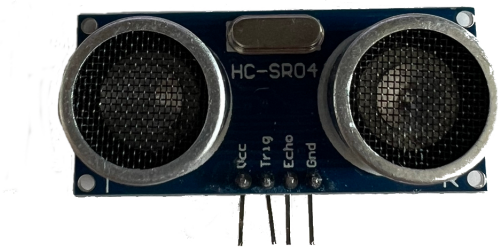
Figur 2.7: Liquid-Crystal skärm 20x4

20x4 LCD-skärmen består av en matris av individuella segment, som tillsammans kan bilda olika tecken, siffror och andra symboler.

Denna typ av LCD-skärm används ofta i applikationer där mer information behöver visas samtidigt på en skärm, som i temperaturmätare, räknare eller annan typ av mätutrustning. skärmen kan också användas för att visa stora textmeddelanden eller för att visa flera parametrar samtidigt.

2.5.4 Ultraljudsgivare

Ultraljudsgivaren är en sensor som mäter avståndet till ett föremål. Ultraljudsgivaren fungerar genom att skicka ultraljudsvågor som reflekteras tillbaka efter att det träffat ett föremål [**?**]. Den data som skickas till Arduinon är tiden det tar för ultraljudvågorna att ta sig fram och tillbaka. Med detta kan användaren mäta avståndet till föremålet. Ultraljudsgivare använder SONAR för att mäta längden till föremålet. SONAR är metoden att mäta avstånd med hjälp av ljudvågor [**?**]. Räckvidden för ultraljudsgivaren är från 2 till 400 cm och det behövs ingen kontakt med föremålet för att fungera. Funktionaliteten är ej påverkad av solljus eller svart material men mjuka material såsom tyg kan påverka funktionaliteten. Ultraljudsgivaren är uppbyggd på två givare där ena skickar ut signaler och den andra tar emot signaler som kan ses i Figur 2.8



Figur 2.8: Ultraljudsgivare

Sensorn använder ljudets reflektion för att mäta tiden mellan ljudvågen som skickats ut och den mottagna ljudvågen. Tiden används för att beräkna avstånden med den konstanta ljudhastigheten som är 340 m/s [**?**]. Med den enkla matematiska ekvationen avstånd = hastighet (340 m/s) * tid, mäter sensorn avståndet.

2.5.5 Vätskepump

En vätskepump till Arduino är en elektronisk komponent som används för att pumpa vätskor i olika applikationer. Vätskepumpen består av en motor som driver en pump, vilket skapar ett tryck som driver vätskan genom en slang eller ett rör.

För att styra vätskepumpen med en Arduino använder man vanligtvis en motorstyrningsmodul. Denna modul gör det möjligt att styra pumpens hastighet och riktning genom att ändra spänningen och polariteten på motorns anslutningar[15]. Genom att använda en sådan modul kan man också använda en extern strömkälla för att driva motorn, vilket gör det möjligt att pumpa vätskor med högre flöde och tryck.

Vätskepumpar används i många olika applikationer, från akvarier och bevattningssystem till medicinska apparater och labbexperiment. De är särskilt användbara i automatiserade system där man behöver reglera flödet av vätska på ett exakt och kontrollerat sätt. Med en vätskepump till Arduino kan man skapa ett automatiserat system som kan pumpa, dosera och reglera vätskor baserat på olika sensorer och kontrollsignalerna från Arduino.

2.5.6 Relämodul

En 5 volts relämodul är en elektronisk komponent som används för att styra elektriska kretsar genom att öppna och stänga en elektrisk kontakt. Komponenten kan ses i Figur 2.9. Relämodulen består av en spole som när den aktiveras skapar ett magnetfält som drar till sig en kontakt som öppnar eller stänger beroende på relämodulens design [16].

Figur 2.9: Relämodul för Arduino med spänningen 5V

En 5 volts (V) relämodul använder en spänning på 5 volt för att aktivera spolen och därmed öppna eller stänga kontakten. Detta gör det enkelt att integrera relämodulen i en mängd olika projekt eftersom 5 volt är en vanlig spänning som är

lättillgänglig från många källor [16]. Relämoduler används ofta i applikationer där man behöver styra högre spännings- eller strömstyrkor än vad en mikrokontroller eller annan lågspänningskrets kan hantera direkt.

Relämoduler kan användas för att styra allt från belysning och motorer till värmeelement och stora industriella maskiner. De är också vanliga i IoT-projekt och hemautomation, där de kan användas för att styra allt från smarta lampor och lås till garageportar och bevattningssystem [16].

2.5.7 Nätaggregat

Nätaggregat används för att försörja tekniska komponenter med spänning samt ström. Det Nätaggregatet som har använts är EL302RT. EL-R serien bidrar till en medelnivå av effekt som kan stiga upp till 125 watt [17]. EL302RT modellen kan spänningen varieras mellan 0-30V. Strömmen kan varieras mellan 0-2 ampere (A) som kan visas på Figur 2.10

Figur 2.10: Nätaggregatet EL302RT

Varje utgång har sin egen av-på knapp som kan användas för att stänga samt sätta på strömförsörjningen. EL302RT modellen skiljer sig ifrån de andra modellerna då modellen har extra logikspännings utgång med en strömkapacitet på 2A. Nätaggregatet har en positiv utgång samt en negativ utgång. De två utgångar som finns i mitten av nätaggregatet kan endast variera mellan 1.5-5V. Den typ av ström som skickas ut från Nätaggregat är likström (DC) och inte växelström (AC).

2.5.8 EasyEDA

EasyEDA är en molnbaserad programvara för kretskortsdesign och kretssimulering som tillåter användare att skapa scheman och kretskort med enkla och intuitiva verktyg. EasyEDA tillhandahåller en mängd olika funktioner och verktyg, inklusive en omfattande bibliotek med komponenter och en integrerad Spice-krets simulator[18]. EasyEDA är gratis att använda, men det finns också möjlighet att uppgradera till en betald version för att få tillgång till ytterligare funktioner och mer utrymme för projekt.

Ett av EasyEDAs mest använda verktyg är dess schemaverktyg, som gör det enkelt att skapa och redigera kretsscheman. Schemaverktyget innehåller många avancerade funktioner, såsom hierarkiska kretsscheman, automatisk anpassning av kabeldragning och stöd för återanvändning av kretsar. EasyEDA har också möjlighet att beställa tryckta kretskort direkt från programvaran[18]. Utöver detta så finns det ett bibliotek där man kan söka på vad som helst och det dyker antingen upp modeller som redan finns tillgängliga på internet eller användarbaserade 2D modeller. Det behövdes ett program där alla komponenter var lättillgängliga. Då de komponenterna vi använder till projektet är väldigt moderna.

3
Metod

Här beskrivs de olika metoder och tillvägagångssätt som kommer användas för projektet. Dessutom diskuteras vilka simuleringar som kommer att utföras samt begränsningar som projektet har.

3.1 Simulation

3.1.1 Systemets funktioner

Arbetet är uppbyggt på 5 starka krav som är bestämda av Veronica Olesen som presenterades i kapitel 1.4.1. Krav ett är att koppla en vattenpump som skall ställas in på önskat flöde och krav två är att det ska gå att mäta vattennivån i tanken. Utöver detta ska det finnas en möjlighet att ställa in en dödtid. Det skall också finnas en möjlighet att styra systemet med analog signal från en PLC. Mät signalen motsvarande vattnets nivå skall också kunna skickas till PLC:n. Det finns också starka önskemål och de är att PID-regulator ska implementeras dessutom skall PID-parametrarna visas och bestämmas med ett användargränssnitt. Systemet ska också kunna köras manuellt med användandet av en vridpotentiometer som kan justera vätskepumpensflöde.

För test samt implementering av de flesta kraven och önskemålen har Tinkercad använts. Tinkercad användes för att ej förstöra komponenter samt att förhandsprogrammera vissa funktioner såsom nivåmätningen. Användargränssnittet skapades också med Tinkercad och användes som en grund för systemet. För att testa systemet delades arbetet upp i två delar. Den första delen är mer gränssnitt baserad och användarbaserad. Andra delen är mer funktionsbaserad beroende på användarens inmatning. Första delen innehåller P, I och D variabler och användaren kan själv bestämma värdena för dessa variabler. Beroende på inmatningen kommer systemet att agera på ett unikt sätt. Utöver detta bestäms ett börvärde för vattennivån som pumpen ska uppnå. Hur systemet agerar är beroende på hur stort integral samt derivataverkan är. De insatta värdena visas för användaren och nivåmätningen påbörjas när värdena är bestämda. Den andra delen består av att bearbeta de inmatade värdena. PID regleringen implementeras med funktionerna som beskrivs i teorin på kapitlet 2.2.4. Vattenpumpen får ett börvärde och med börvärdet kan pumpen manipuleras till att antingen pumpa ut vatten eller inte.

3.1.2 Vattentankens uppbyggnad

För att implementera systemets funktioner krävs elektriska komponenter. För att implementera krav 2 behövs en ultraljudsgivare. Med hjälp av ultraljudsgivarens ljudvågor som skickats ut kan avståndet från ultraljudsgivaren till vattenbehållaren beräknas. Avståndet från sensorn till behållaren skickas till en LCD-skärm som visar resultatet. Med LCD-skärmen får användaren en god visuell beskrivning av hur mycket vatten som finns i behållaren. Utöver det finns det två potentiometrar för att användaren ska kunna bestämma I och D värdet. Potentiometern kan vridas i en 180 graders vinkel och beroende på hur stor vridningen är desto större värde. Värdet kan användaren läsa av i LCD-skärmen. För bestämmelse av börvärdet kommer det finnas en annan potentiometer och användaren kommer också att kunna lösa av detta på LCD-skärmen. Krav 3 är att användaren skall kunna bestämma fördröjningen med en potentiometer. Systemet fungerar på det viset att det körs om och om igen i en loop. Värdet på fördröjningen ska variera mellan 0 och 3 sekunder beroende på inmatningen av potentiometern. Fördröjningen kommer bidra till att vätskepumpen fördröjs med antalet inmatade sekunder. Detta är viktigt för att kunna få en inblick kring hur fördröjningen kan påverka systemet. För att användaren ska kunna växla till manuell styrning finns det en switch som när den är på antyder att systemet styrs manuellt och detta visas också på LCD-skärmen. Vätskepumpens varvtal ska bestämmas via en analog signal som skickas ifrån PLC. Signalen som skickas från PLC ska avläsas i Arduinon och Arduinon ska skicka data på vattennivån från ultraljudsgivaren till PLC:n.

3.1.3 Bevis på koncept

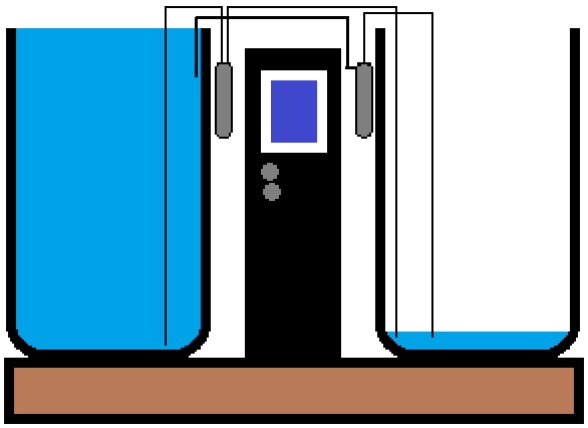
Arbetet var tänkt i början att implementeras till chalmerister men vid samtal med Veronica bestämdes att arbetet skall begränsas till ett bevis på koncept eller på engelska proof of concept arbete. Detta betyder att arbetet går ut på att bevisa att detta är möjligt att skapas med hjälp av Arduino och PLC, sedan om läraren vill själv implementera systemet för att användas i Chalmers så ska det vara möjligt med rätt komponenter.

3.2 Konstruktion

3.2.1 Ritning

Efter att vi hade implementerat all funktionalitet med hjälp av Arduino, genomförde vi en designprocess där vi skapade olika modeller för att visualisera det slutliga systemets utseende. För att återskapa strukturen av systemet använde vi ett antal träplankor som sammanfogades för att skapa en korsliknande konstruktion som höll upp de två vattentankarna. Systemets layout och struktur kan ses i figur 3.1.

Den valda designen för projektet involverade användningen av två brädor. Den understa brädan användes för att ge stabilitet åt systemet och förhindra att komponenter eller delar rörde sig. En svart planka placerades i mitten av systemet och



Figur 3.1: PID-Regulator Blockschema

fungerade som en stödstruktur för de två LCD-skärmarna och potentiometrarna i systemet. Genom att implementera denna design blev systemet helt komplett och redo för användning.

3.2.2 Elektrisk koppling

För att bygga upp systemet så krävs det både mjuk- och hårdvara. För att all hårdvara ska funktionera samtidigt så behövdes det ett kopplingsdäck. Kopplingsdäcket fanns där för att, kunna reglera spänningen samt att koppla komponenterna. Kopplingsdäcket fungerade som plattform för att bygga upp systemet, med Arduinon som central enhet för att styra och ansluta till olika spänningar.

Systemet består av flera komponenter som kräver en 5V spänningsmatning. Dessa komponenter inkluderar ultraljudssensorn, LCD-skärmen, relämodulen och Arduino. Den enda delen som krävde reglering av spänningen var vattenpumparna eftersom deras varvtal styrdes av spänningen. Regleringen av spänningen till vattenpumparna genomfördes för att uppfylla Veronicas specifika krav.

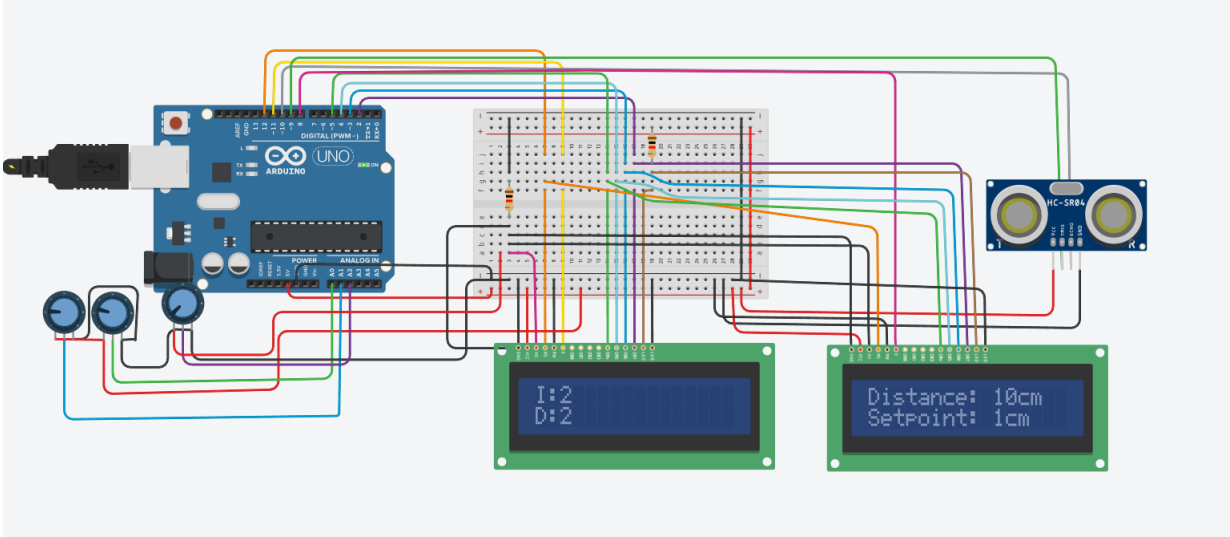
4

Genomförande

Här beskrivs hur arbetet har genomförts i en logisk ordning. Ordningen för genomförandet går hand i hand med arbetets gång och alla upptäckelser beskrivs. Utöver det beskrivs hur problemen har lösts och med vilka programvaror och komponenter.

4.1 Tinkercad

För att skapa en grund på arbetet användes Tinkercad som nämnt i metodkapitlet. Tinkercad designen är inte den fullständiga designen och är endast ett utkast till den verkliga designen. Det finns flera tekniska komponenter som inte finns tillgängliga på Tinkercad såsom vätskepumpen och därför begränsades designen. Kretsen innehåller en Arduino UNO, en bredboard, två LCD-skärmar 16x2, tre Potentiometer, två 1k Ohm resistorer och en ultraljudsgivare som visas på figur 4.1.



Figur 4.1: Tinkercad-krets som består av 3 potentiometrar 2 LCD-skärmar och en ultraljudsgivare. Figuren visar hur alla komponenter är anslutna tillsammans för att uppnå en godtycklig funktionalitet.

4.1.1 Anslutning av Potentiometer

Potentiometern innehåller totalt 3 pins som ska kopplas för att potentiometern ska fungera. Den första terminalen är ansluten till en spänningskälla och den spänningskällan finns hos Arduinos 5V utgång. Den andra terminalen är ansluten till Arduinos jord. Då flera komponenter behöver anslutas till Arduinos jord samt 5V utgång har de kopplats till en bredboard vilket bidrar till en smidigare krets. De två terminalerna på potentiometrarna är anslutna till samma punkt för att skapa en mer stilren krets som sedan skickas vidare till bredboarden. Det som skiljer potentiometrarna åt är den så kallade Wiper. Wiper är det som är inuti potentiometern som användaren kan rotera och den är ansluten till en analog pinne på Arduinon. De analoga pinnarna som har valts att användas är A0,A1 och A2. Potentiometers syfte är att ändra på I-verkan,D-verkan samt bestämma en setpoint som vattnet skall stanna på.

4.1.2 Anslutning av LCD-skärmen

LCD-skärmen har totalt 16 pinnar och utav de 16 är det 12 som är ansluta som kan ses på figur 4.1. Den första LCD-skärmen används för att visa hur stort I samt D verkan är. Med hjälp av de två första potentiometrarna kan värdet ändras beroende på rotationen och det värdet visas upp på skärmen. Den andra LCD-skärmen visar avståndet som ultraljudsgivaren mätt upp samt hur stor setpointen är.

Anslutningen för de två LCD-skärmarna är exakt samma förutom en pinne som skiljer dem åt. Den första pinnen av LCD:n ska anslutas till ground (GND). Pinne två ansluts för att försörja skärmen med spänning. Tredje pinnen av LCD-skärmen är ansluten till en 1kΩ resistor och behövs för att reglera skillnaden på skärmen. Pinne fyra är ansluten till Arduinon och används för att bestämma skärmens läge och en nolla indikerar data läge och etta kommandoläge. Detta behövs för att bestämma kontrasten på skärmen, sätta på och stänga av skärmen eller att välja vilken rad som ska användas på skärmen. Den femte pinnen är ansluten direkt till jord och används bestämma om LCD-skärmen ska skriva eller läsa av. En nolla indikerar att LCD-skärmen ska skriva och en etta indikerar att den i stället ska läsa av. Pin 6 är den enda pinnen som skiljer på båda LCD-skärmar. På den första LCD-skärmen är Pin 6 ansluten till digital pin 12 på Arduinon och det andra skärmen är ansluten till digital pin 8. Det pin 6 gör är att den utför läs/skrivprocessen av LCD-skärmen och det är med hjälp av den pinnen man kan sära på LCD-skärmarna.

Pin 7 till 14 används för att skicka data till skärmen och alla pins ska inte vara anslutna, de är anslutna i så kallad 4-wire mode. Man kan också ansluta LCD-skärmen i en 2-wire mode och 8-wire mode men det rekommenderade är 4-wire mode och det är så den är också ansluten på Tinkercad. Med 4 wire mode är det endast 4 pinnar som är anslutna till Arduinon och de är pin 11 till 14. Pin 15 används för att ge spänning till LED och är därför ansluten till spänningskällan. Pin 16 är jorden för LCD-skärmens LED.

4.1.3 Ultraljudsgivarens Anslutning

ultraljudsgivaren har totalt 4 pinnar som behövs för att den ska fungera. På sensorn finns det en markering som indikerar vilka pinnar som är gjorda för VCC samt GND. VCC är ansluten till Arduinos 5V spänningskälla såsom de andra komponenterna. GND är ansluten till Arduinos ground. Trig pinnen är ansluten till Arduinos digitala pin 9. Trig pinnens anslutning krävs för att skicka ut en puls som kommer att reflekteras mot de objektet som pulsen träffar. Echo-pinnen är ansluten till den digitala pinnen 10. Echo-pinnen producerar en puls när den reflekterade pulsen är mottagen som trig pinnen skickat ut. Pulstiden krävs som beskrivet på den tekniska bakgrunden för att mäta avståndet mellan ultraljudsgivaren och objektet. Echo-pinnen är satt till HIGH när signalen skickas ut och när signalen kommer tillbaka sätts echo-pinnen till LOW.

4.1.4 Tinkercad Programmering

För att få alla komponenter att fungera väl och tillsammans krävs det programmering. Med Tinkercad skapades den grundläggande programkoden som kommer att utvecklas mer under arbetets gång. Programkoden är uppdelad i två stora delar. Första delen är att deklarera globala variabler samt installationen av de komponenter som ska användas. Den andra delen av koden består att implementera de olika funktionerna för komponenterna som har installerats i systemet. Nedanför visas hur den grundläggande installationen för arbetet skapades.

```
// inkludera bibliotekskoden:
#include <LiquidCrystal.h>

//  Initiera biblioteket med sifferkoderna for granssnitten.
LiquidCrystal lcd(12, 11, 5, 4, 3, 2); //Pinsen for den forsta LCD
    displayen
LiquidCrystal lcd2(12, 8, 5, 4, 3, 2); //Pinsen for den andra LCD
    displayen

// Globala Variabler

int pid_I = 0; // PID I varde
int pid_D = 0; // PID D varde
int SETPOINT = 0; // Setpoint varde
const int trigPin = 9; //Ultrasonic Trig
const int echoPin = 10; //Ultrasonic Echo
long duration; //Tiden for ultrasonic
int distanceCm; // Vattennivan i CM


// Standard Setup Function
void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  // Stall in antalet kolumner och rader for LCD-skarmen:
  lcd.begin(16, 2);
```

```
  lcd2.begin(20, 4); // finns inte 20x4 skarm pa tinkercad men
      finns i verkligheten
  // Rensa LCD-skarmen.
  lcd.clear();
  lcd2.clear();

}
```

För att ändra på LCD-displayen med hjälp av Arduino krävs användningen av Liquid Crystal-biblioteket. Det första steget i koden är att initiera biblioteket. Därefter definieras de två LCD-displayerna, vid namn LCD och LCD2 för att skilja dem åt och kunna anropa dem med dessa namn.

Som tidigare beskrivet är alla pinnar på LCD-displayen likadana förutom läs/skriv-pinnen, som är ansluten till pinne 6. Detta visas i LCD-displayens installation där pinne 6 för den första LCD-skärmen är ansluten till den digitala pinne 11 och för den andra LCD-skärmen är den ansluten till digitala pinne 8.

Därefter initieras de globala variablerna. De första globala variablerna används för att ändra inställningarna för I och D-verkan för PID-regleringen. Setpoint-variabeln behövs för att systemet ska veta när vattenpumpen ska stängas av och vid vilken höjd den ska stanna.

Därefter definieras pinnarna för ultraljudsgivaren, vilka är trig- och echo-pinnarna och är anslutna till digitala pinne 9 respektive 10.

Variabeln ”duration” används för att mäta tiden det tar för ultraljudsgivaren att träffa ett objekt. Den är av typen ”long”, vilket är samma som ”integer” men används för att kunna hantera större heltal. Slutligen finns den sista globala variabeln ”distanceCM” som kommer att användas för att beräkna vattenhöjden.

Funktionen ”void setup” används för att initiera variablerna och pinnarna. ”Serial.begin(9600)” behövs för att Arduino ska vara redo att utbyta meddelanden med en serieövervakare vid en datahastighet på 9600 bitar per sekund. Serieövervakaren används för att utföra kontroller att rätt data skrivs ut.

Funktionen ”pinMode” används för att konfigurera pinnarna för ultraljudsgivaren. I detta fall indikerar det att den globala variabeln ”trigPin” är en utgång och ”echoPin” är en ingång. Det betyder att trig-pinnen kommer att sända ut ljudsignalen och echo-pinnen tar emot den reflekterade ljudsignalen.

”Lcd.begin” startar LCD-skärmen och beskriver LCD-skärmens upplösning. I det här fallet är det angivet som (16,2), vilket betyder 16 kolumner och 2 rader. På Tinkercad finns endast 16x2 LCD-skärmar tillgängliga, men i det verkliga systemet kommer både 16x2 och 20x4 LCD-skärmar att användas. ”Lcd.clear” används för att rensa den tidigare data från skärmarna.

```
// Visa vardena pa LCD skarmarna
void displayValues() {
```

```
  lcd.setCursor(0, 0); // Indikerar att det ska printas pa forsta
     raden av LCD skarmen
  lcd.print("I:");
  lcd.print(pid_I);
  lcd.print("   ");

  lcd.setCursor(0, 1); // Indikerar att det ska printas pa andra
     raden av LCD skarmen
  lcd.print("D:");
  lcd.print(pid_D);
  lcd.print("   ");

  lcd2.setCursor(0, 0);
  lcd2.print("Distance: "); // Skriver ut strangen "Distance" pa
     LCD-skarmen.
  lcd2.print(ProcentMeasured); // Skriver ut avstandsvardet fran
     sensorn.
  lcd2.print("%");
  lcd2.print("   ");

  lcd2.setCursor(0, 1);
  lcd2.print("Setpoint: "); // Skriver ut strangen "Setpoint:" pa
     LCD-skarmen.
  lcd2.print(SETPOINT); // Skriver ut setpointen
  lcd2.print("%");
  lcd2.print("   ");
}
```

För att LCD-skärmarna ska visa data har en funktion skapats vid namn ”displayValues”. Kommandot ”setCursor” indikerar vilken kolumn och rad data ska skrivas på. ”Lcd.setCursor(0,0)” betyder att data ska börja skrivas på första kolumnen och första raden på skärm ett.

Citattecknet i ”lcd.print” indikerar att en sträng ska skrivas ut. I detta fall är det strängen ”I:” som visas för att visa värdet på I-verkan. När I eller D värdet minskar från till exempel 10 till 9 kommer nollan i slutet att stå kvar för skärmen har inte återställts. För att undvika detta har ”lcd.print(” ”)” vilket skapar ett mellanrum där nya värden kan sättas in och återställer gamla värden.

Den andra LCD-skärmen ska visa data från ultraljudsgivaren och den setpoint som är bestämd med potentiometern. Principen för den andra skärmen är exakt densamma som för den första.

```
// Huvudprogrammet Loop
void loop() {
  int a0 = analogRead(A0); // Las av vardet fran potentiometern.
  int a1 = analogRead(A1);
  int a2 = analogRead(A2);
  // Justera vardena oberoende av varandra.
  pid_I = map(a0, 1023, 0, 0, 10); // Vardet pa potentiometerns
     vridning kan variera fran 0 till 10
  pid_D = map(a1, 1023, 0, 0, 10); // Vardet pa potentiometerns
     vridning kan variera fran 0 till 10
  SETPOINT = (map(a2, 1023, 0, 0, 100)); // Vardet pa
```

```
    potentiometerns vridning kan variera fran 0 till 100
  // Visa de uppdaterade vardena.
  displayValues();

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distanceCm= 300-(duration*0.034/2);
}
```

Huvudprogrammet innehåller en funktion som kallas ”loop” och körs flera gånger beroende på simulationens längd. I början av huvudprogrammet har tre variabler skapats med namnen ”a0”, ”a1” och ”a2”. Dessa står för analog pinne noll, ett och två, vilket representerar anslutningen för potentiometern.

Funktionen ”analogRead” används för att läsa in ett värde mellan 0 och 1023, vilket motsvarar spänningen från 0 till 5V. Detta gör det möjligt att använda funktionen ”map” för att konvertera värdet till det önskade intervallet. För I- och D-verkan ska användaren kunna justera värdet mellan 0 och 10, och med hjälp av ”map”-funktionen görs denna konvertering. För setpoint ska användaren kunna justera värdet mellan 0 och 25.

Funktionen ”displayValues” anropas för att indikera att resultaten ska visas på LCD-skärmen. ”digitalWrite(trigPin, LOW)” skickar en signal till ultraljudsgivarens trig-pinne för att sätta den till lågt värde och bekräfta att ultraljudsgivaren är avstängd. Efter en kort fördröjning på 2 mikrosekunder säkerställs att triggen är avstängd. Triggen på ultraljudsgivaren aktiveras sedan i 10 mikrosekunder för att skicka ut en signal, och stängs sedan av. Echo-pinnen sätts till högt värde för att ta emot signalen från triggen och möjliggör beräkning av avståndet mellan ultraljudsgivaren och objektet.

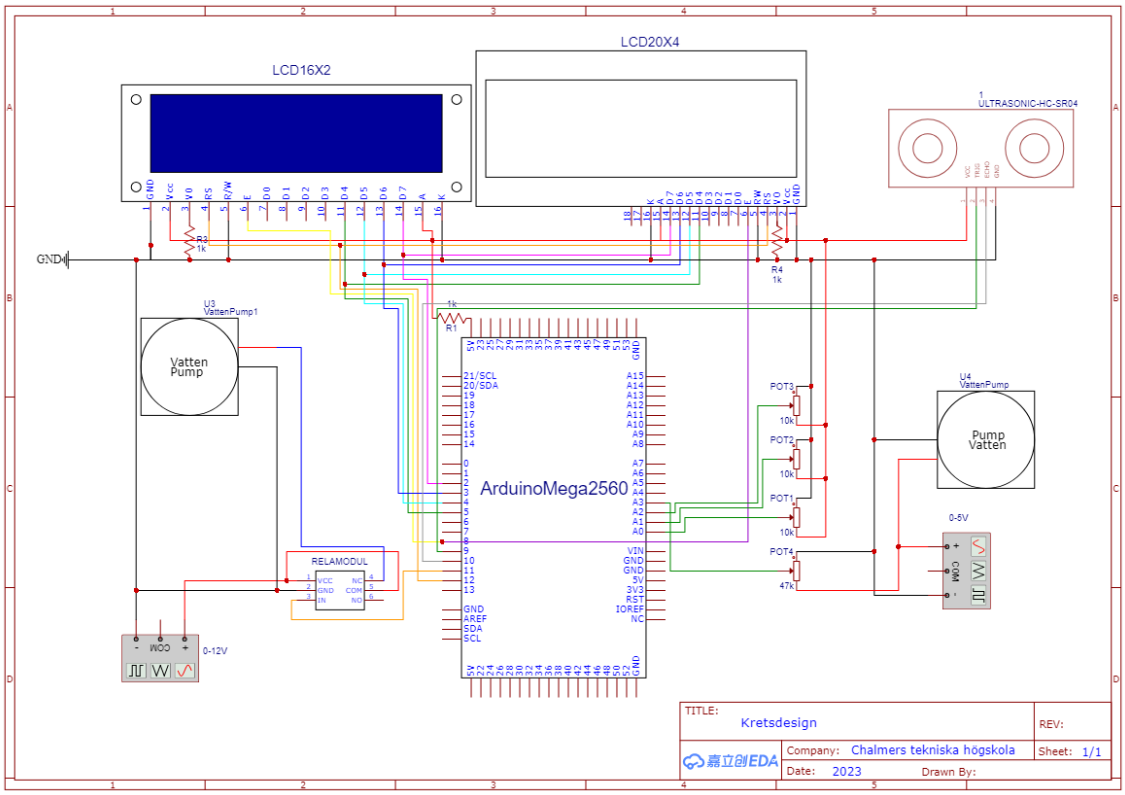
Avståndet beräknas genom att mäta tiden för signalen och multiplicera den med 0,034 dividerat med 2. Detta görs eftersom ljudets hastighet är ungefär 343 m/s, och eko-signalen mäter avståndet två gånger (fram och tillbaka). Avståndet subtraheras sedan med 300, vilket är ett exempel, eftersom i det verkliga systemet skulle det vara subtraherat med höjden på cylindern. Detta görs för att när vattennivån stiger kommer avståndet att öka och visas på LCD-skärmen.

4.2 Kretsdesign

Denna sektion innehåller uppbyggnaden av kretsen och de delar som tog mest tid gällande hårdvara och krävde utförligt arbete. För att förverkliga kretsen krävdes inte bara omfattande arbete, utan också flera simuleringar och tester i verkliga miljöer. Genom att designa kretsen i ett CAD-program skapades en blåkopia för slutprodukten. Detta säkerställde att kretsen fanns som en konkret representation

av den avsedda designen.

När alla delar var på plats och kunde assembleras på kretskortet så var hårdvaran färdigkopplad. Det behövdes då ett program som kunde användas för att rita kretsschema. EasyEda var det bästa alternativet då programvaran erbjuder de 2D-modeller som behövdes för kretsen till projektet. Kretsschemat i figur 4.2 återspeglar det system som byggdes upp för projektet. EasyEDA var programmet som hjälpte till att designa kretsen på bästa möjliga sätt.



Figur 4.2: Kretsdesign från EasyEDA

Kretsen byggdes upp för hand i verkligheten efter design med hjälp av EasyEDA. Hårdvaran funktionerar med hjälp av Arduinon och kopplingen gjordes tidigare med hjälp av Tinkercad. Med hjälp av ett stort kretskort så kopplades LCD-displayen, potentiometrar, spänningstillförsel från spänningsaggregatet, Arduino Mega 2560, relämodul, och 2 st vattenpumpar.

4.2.1 Relämodul Anslutning

För att implementera PID-reglering användes en relämodul i kretsen. Relämodulen är ansluten till pin 11 på Arduino eftersom Arduino-hårdvaran har en begränsning på 5V. För att överkomma detta krävs en relämodul som är kopplad till en extern spänningskälla på 12V. Detta krävs eftersom en av vattenpumparna i systemet kan hantera spänningsnivåer från 0 till 12V. I programkoden samverkar PID-regleringen

med relämodulen för att reglera avstängning eller aktivering av spänningsförsörjningen till den aktuella vattenpumpen.

4.2.2 Vattenpump Anslutning

För att implementera PID-reglering i vattentanken användes två olika vattenpumpar med olika flödesstyrkor. En vattenpump med en flödesstyrka på 2.2 L/min användes för att reglera vattennivån med hjälp av PID-reglering. Den andra vattenpumpen, med en styrka på 2.8 L/min, användes som en konstant störning för att skapa över- och underskjutningar i systemet.

För att åstadkomma över- och underskjutningar krävs en konstant störning i systemet. Genom att använda vattenpumpen med en styrka på 2.8 L/min kunde vatten pumpas ut från vattentanken och skapa en konstant störning i PID-ekvationen. Systemet arbetar sedan för att minimera felet och reglera vattennivån så exakt som möjligt.

Med hjälp av programkoden, relämodulen och en ultraljudsgivare byggdes ett system där ultraljudsgivaren mäter vattennivån samtidigt som pumparna ökar och minskar flödet. Feedback från sensorn används för att avgöra när pumpen ska vara igång och när den ska stängas av, vilket möjliggör en noggrann reglering av vattennivån i tanken.

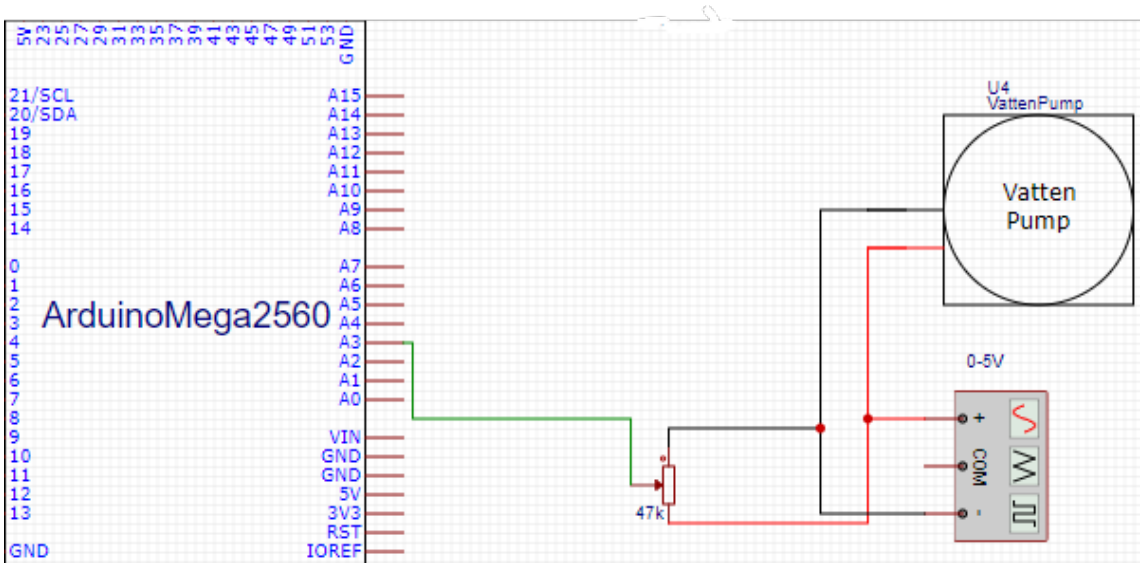
För att framgångsrikt ansluta de två vattenpumparna behövde de separata spänningskällor. Vattenpumpen med en styrka på 2.8 L/min var direkt ansluten till spänningsaggregatet med en konstant spänning på 5V. Detta valdes för att säkerställa att resultaten inte varierade mycket. Den andra vattenpumpen, som var huvudpumpen i reglersystemet, anslöts till en spänning på 3-12V beroende på inställningen av spänningsaggregatet.

4.2.3 Potentiometer Resistor Anslutning

Potentiometern kan kopplas enligt många konfigurationer. Potentiometern kan kopplas antingen i en 2-tråds konfiguration eller en 3-tråds. Till vattenpumpen så behövdes det en potentiometer som kunde agera som en resistor mellan motorn och spänningskällan. Potentiometern behövde mer motstånd än de som beställdes in på $10k\Omega$. Ett försök gjordes till en början med resistorn på $10k\Omega$ och det resulterade i en bränd resistor. Det berodde på att strömmen var för hög genom potentiometern. För att undvika vidare brända resistorer så ökades motståndet 4.7 gånger mer. Ett vidare test gjordes med en potentiometer med en motstånd på $47k\Omega$ och resulterade i ett lyckat försök.

Då ett av kraven för systemet var att man skulle kunna ställa in önskat varvtal/flöde på en av vattenpumparna. Det skapades genom att koppla vattenpumpen med styrkan 2.8L/min till potentiometern med $47k\Omega$. Potentiometern, vattenpumpen och spänningsaggregatet var kopplat i ett så kallat 3-tråds konfiguration. Där

potentiometern funkar som en resistans mellan motor och spänning. Vrider man potentiometern åt full potential så kommer spänningstillförseln till motorn att minska tills den avbryts helt. I figur 4.3 nedanför kan man se konfigurationen med potentiometern.



Figur 4.3: Kretsdesign för 3-tråds konfiguration

4.3 Programkod

Den fullständiga programkoden bakom styrningen av vattentank-systemet ges i Appendix A. Programkoden förklaras mer i detalj där små delar plockas ut och logiken bakom beskrivs. Mycket av koden är lik Tinkercad-koden då den har använts som en grund för att skapa den fullständiga logiken för vattentanken.

4.3.1 Void Setup

Vid initieringen av programkoden för självaste simulationen i verkligheten behövdes fler initieringar i ”void setup()”. Detta var på grund av att Tinkercad inte kunde simulera en vattenpump. Koden för vattenpumpen fick göras separat från simulationen.

```
void setup() {
  Serial.begin(9600);
  pinMode(pump1, OUTPUT);  // variant low/high
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  lcd2.begin(20, 4);
  // Clear the LCD.
  lcd.clear();
  lcd2 .clear();
```

```
  pinMode(Auto1, INPUT_PULLUP);
  pinMode(Manual1, INPUT_PULLUP);
}
}
```

Allt förutom koden nedanför har förklarats i sektion 4.1

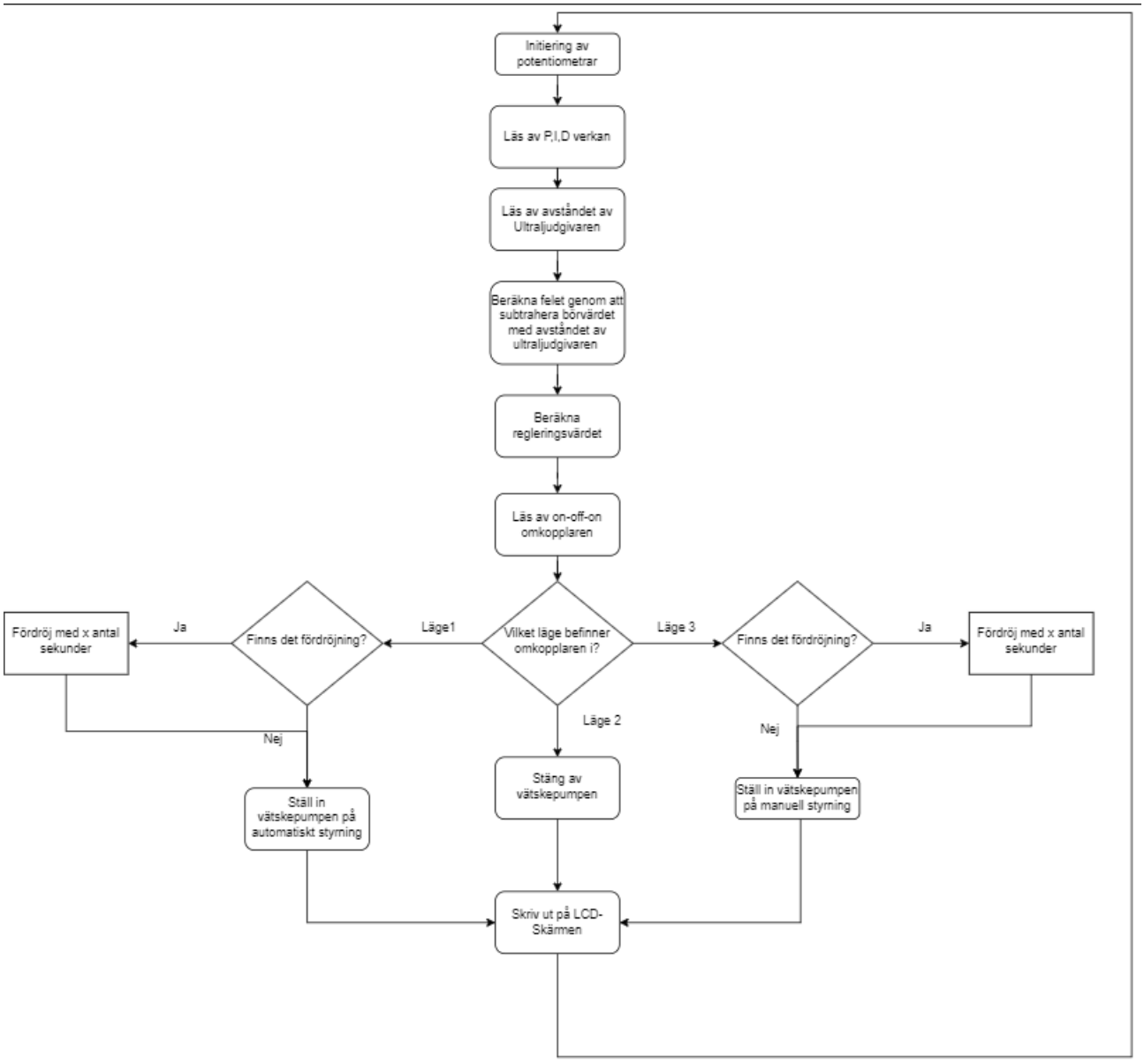
```
{
  pinMode(Auto1, INPUT_PULLUP);
  pinMode(Manual1, INPUT_PULLUP);
}
```

I denna specifika kod ovanför används funktionen ’pinMode()’ för att sätta läget för två pinnar, ’Auto1’ och ’Manual1’, till INPUT PULLUP. Läget INPUT PULLUP aktiverar den interna pullup-resistor på de angivna pinnarna, vilket säkerställer att de är högspända (i ett logiskt HIGH-läge) som standard när ingen extern ingång är ansluten.

För att använda denna kod behöver man definiera konstanterna ’Auto1’ och ’Manual1’. De definitioner som användes för projektet kan ses i koden nedanför.

```
//SWITCH
const int Auto1 = 52;
const int Manual1 = 50;
```

4.3.2 Huvudprogrammet



Figur 4.4: Flödesdiagram av hur huvudloopen fungerar

4.3.2.1 PID-Reglering logik

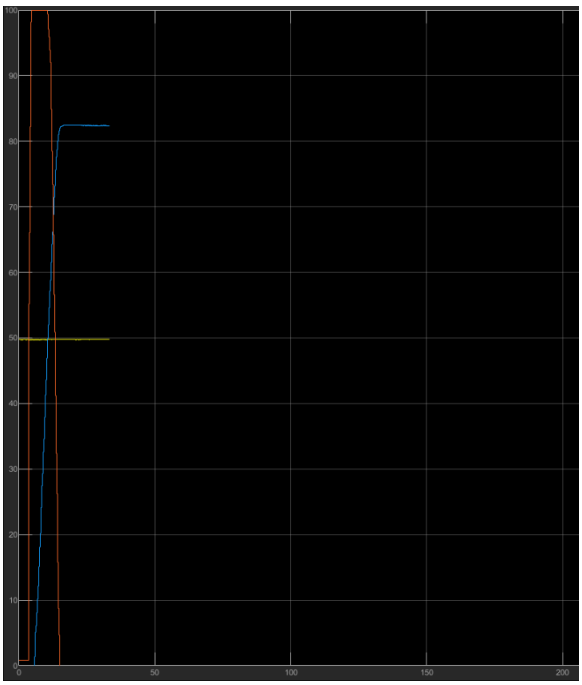
PID regleringen fungerar på samma sätt som beskrivet i kapitel 2.2.4 Inmatningen till systemet är vattennivån som bestäms av ultraljudsgivaren. Felet är beräknat genom att användarens börvärde subtraheras med uppmätta värdet av ultraljudsgivaren. Integralvärdet är beräknat genom att summera det tidigare integralvärdet samt felet på systemet. Derivata-värdet är bestämt genom att subtrahera det nuvarande felet på systemet med det tidigare uppmätta felet. Utgångsvärdet blir en summering mellan felet, integralvärdet samt det derivata värdet. Felet är multiplicerat med användarens inmatning av P-verkan. Ki är användarens inmatning på I-verkan och Kd-inmatning på D-verkan.

5
Resultat

Här presenteras resultaten av arbetet. De resultat som tagits från det tidigare existerande systemet presenteras för att ha något att jämföra med det nya systemet.

5.1 Simulationsresultat - Verkliga systemet

För att uppnå en bra jämförelse testades det verkliga vattentanksystemet som finns på Chalmers Campus Lindholmen. För det verkliga systemet bestämdes börvärdet till 50 % av tankens totala höjd som är 60 cm. De två olika testerna som gjordes var att testa systemet utan störning och med störning med samma PID-verkan. PID-verkan för testerna var bestämt till 1 för alla parametrar för att få en god jämförelse. Två stegsvars plottar skapades för att kunna se hur systemet påverkas. Vattentanken i Lindholmen kan anslutas till programmet MATLAB för att skapa stegsvars plottar. Det första testet som utfördes var mätningen utan störning som visas nedan på figuren 5.1.

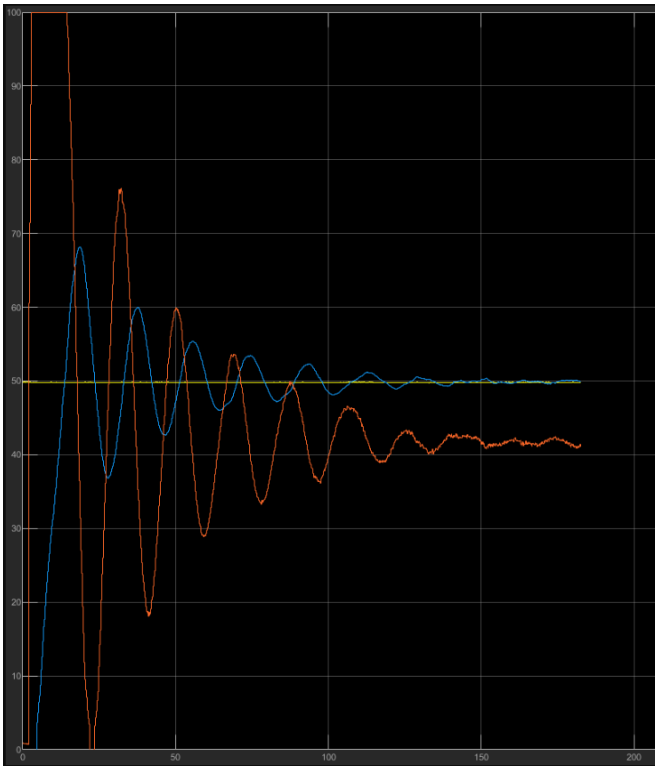


Figur 5.1: PID-reglering utan störning på verkliga systemet där vattenhöjden,styrsignalen samt börvärdet visas

Den blåa signalen är mätvärdet, den gula signalen är börvärdet och den orangea

signalen är systemets styrsignal som motsvarar vattenpumpens styrka. Resultatet av stegsvaret visar att börvärdet är mindre än det uppmätta värdet. Börvärdet är bestämt till 50 % som nämnt tidigare och det uppmätta värdet enligt 5.1 är cirka 83 %. Överskjutningen beror på inställningen av PID-parametrarna. Det kan bero på långsam integralåterkoppling, alltså lågt I värde eller för liten dämpning på systemet. För att få bästa resultat kan en av justeringsmetoderna som nämns på 2.3 användas.

På det verkliga systemet finns det en ventil som kan öppnas för att skapa en störning på systemet. Ventilen gör att vattnet börjar rinna ner samtidigt som vattenpumpen är i gång vilket skapar en störning på systemet. Stegsvaret för testet med störning visas på figuren 5.2.



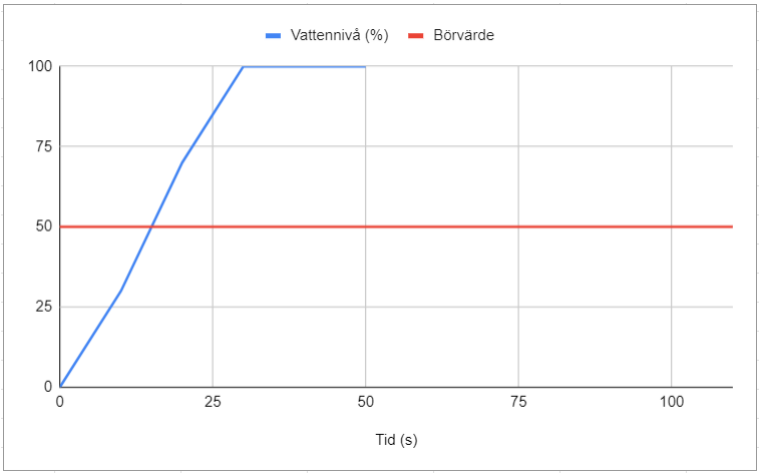
Figur 5.2: PID-reglering med störning på verkliga systemet där vattenhöjden, styrsignalen samt börvärdet visas

Resultatet av testet bidrog till flera underskjutningar och överskjutningar. Första överskjutningen låg på cirka 68 % och underskjutningen på 38 %. Här beror undersamt överskjutningarna på inställningen av PID-parametrarna samt systemet störning. Det tog cirka 160 sekunder för systemet att stabilisera.

5.2 Simulationsresultat - Proof of concept systemet

Från proof of concept systemet kan vi utläsa data från Arduino IDE:s data plotter. För att realisera stegsvaret i ett linjediagram behövde data extraheras till Excel för

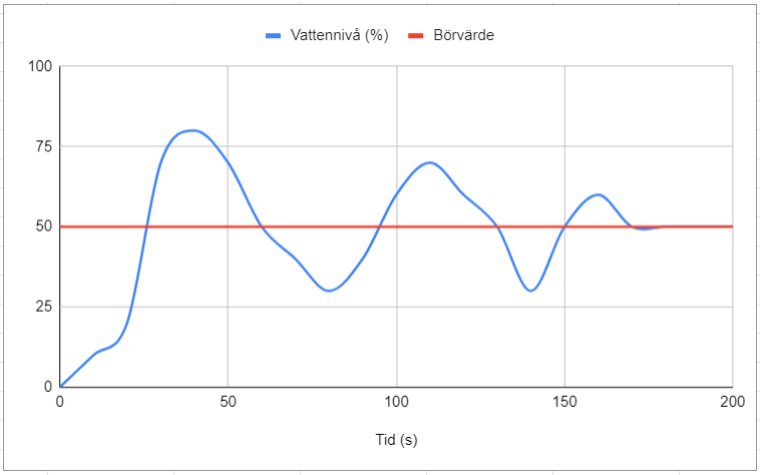
att skapa stegsvaret. Nedanför i figur 5.3. är det första testresultatet från systemet utan störning.



Figur 5.3: PID-reglering utan störning på proof of concept systemet

Börvärdet som var satt för systemet ovanför var 50 %. Som resultatet visar så har det uppstått en överskjutning i det systemet. Vattenpumpen som användes för att reglera systemet var inställd på 10 V. Överskjutningen beror på att PID-verkan var i gång samtidigt som det inte fanns störning på systemet.

Systemet testat för reglering med och utan störning. Nedanför i figur 5.4 visas resultatet för systemet då en störning på 5 V lades på systemet. Börvärdet för det systemet var insatt på 50 % samma som tidigare simuleringsresultat. Det tog ungefär 170 sekunder för systemet att reglera sig med hjälp av parametrarna P: 1, I: 1, och D:1. Vattenpumpen som används för att reglera systemet var insatt på 10 V som tidigare försök.



Figur 5.4: PID-reglering med störning på proof of concept systemet

Grafen ovanför visar hur systemet över-/underskjuter en hel del tills den till slut reglerat höjden på vattennivån till 50 %.

5.3 Simulationsjämförelse

Det verkliga systemet är mycket större än proof of concept systemet. Höjden på proof of concept systemet är 20 cm medan det verkliga systemet har en höjd på 60 cm. För att göra jämförelsen bestämdes ett börvärde på 50 % av tankens totala höjd. Däremot proof of concept systemet använder endast 10 cm för simulationer och inte 20 cm. Detta för att kunna få en bättre överblick på funktionaliteten. En jämförelse skapas då 50 % på börvärdet resulterar i 10 cm på proof of concept systemet och 30 cm på det verkliga systemet. Testerna visade att resultatet på testet utan störning hade olikheter. Concept systemet hade en större överskjutning som låg på 100 % vilket motsvarar hela tankens höjd medan det verkliga systemet hade en överskjutning på cirka 83 %. Denna skillnad beror på vattenpumpens styrka och de tekniska komponenter som har använts.

Verkliga systemets körtid för att stabilisera var cirka 170 sekunder vilket var nästan exakt samma som concept systemet. Däremot visade verkliga systemet till många fler över- och underskjutningar. Totalt som kan synas på figuren 5.2 har 6 överskjutningar skett till stabiliserad nivå har uppnåtts. Concept systemet har endast 3 överskjutningar skett tills det stabiliserat vilket bidrar till ett mer stabilt system. Däremot varade överskjutningarna i mycket längre tidsperioder detta beror på att kraften av störningen var svagare än det i det verkliga systemet. Men som helhet utför systemet samma funktionalitet av det verkliga systemet med små variationer. Dessa variationer är beroende av komponent olikheterna mellan de två systemen. Ett sådant exempel är olikheterna av vätskepumparna som har använts för respektive system.

6
Diskussion

I detta kapitel kommer vi diskutera de komplikationer som har uppstått under arbetets gång. Dessutom kommer vi att gå in i mer detalj kring de missade kraven samt ändringarna som utförts. Vi kommer även att diskutera hur framtiden hade sett ut om vi kunde fortsätta mer med arbetet och haft mer tid. Arbetet kommer också att diskuteras ur hållbarhetens aspekter.

6.1 Komplikationer

Det var tänkt att vårt arbete skulle implementeras på Chalmers labbsalar, men det ändrades ungefär 1 månad efter att vi hade börjat. Det hade varit misskommunikation mellan Sakib och Veronica och bidrog till att arbetet blev lite försenat.

Arbetet blev i stället ett proof of concept arbete, vilket innebär att vi endast ska undersöka om det går att implementera ett system med Arduino. Detta gjorde oss lite förvirrade kring om vi endast ska skapa en simulation eller implementera det i verkligheten. Vi bestämde oss för att implementera det i verkligheten då Tinkercad inte hade någon vattenpump.

Om vi kunde göra om arbetet från början så att vi gjort flera saker olika. Vi hade inte skapat planeringen direkt efter kravlistan och i stället forskat lite innan och tänkt ut vad som hade varit mest tidskrävande och lagt de längre fram i planeringen.

Vid en av testerna fick 20x4 LCD-skärmen för mycket spänning och den brann upp. Detta gjorde att vi fick vänta tills den nya LCD-skärmen var beställd. Vi hade också kunnat göra arbetet mindre komponentberoende. Eftersom vi behövde vänta på att komponenter levererades. Vi hade kanske kunnat hitta en annan online simulering än Tinkercad som erbjuder fler komponenter som vattenpump och relämodul.

Utöver dessa komplikationer har arbetet gått bra och vi har följt planeringen. Vi har också gjort veckovisa rapporter som har skickats till Lena så att hon har koll på vad som sker med projektet. De veckovisa rapporterna som vi skickat till Lena hjälpte oss med att veta vart vi ligger till

6.2 Uppfyllda krav

Totalt ställdes 8 krav varav 5 krav var starka. Arbetet som har utförts har lyckats uppfylla alla krav förutom ett som kommer att diskuteras i 6.3. Concept systemet kan styra vattenpumpen via spänningsaggregatet samt Arduinon. Dessutom kan mätningen av vattennivån mätas med ultraljudsgivaren. Användaren har också en möjlighet att ställa in en död tid mellan 0–3 sekunder via en potentiometer som beskrivet i kapitel 4. Veronica ställde kravet att styra pumpens varvtal från PLC samt skicka analog signal på vattennivån till PLC:n som beskrivet i 1.4. Detta har uppfyllts men med lite variationer. Implementationen av PLC ansågs ej nödvändig då spänningsaggregatet utför samma funktionalitet och därför har detta krav ersatts med att styra via spännings aggregaten. Innan ändringen gjordes diskuterades ändringen med Veronica och hon förstod principen. Vidare diskussioner kring varför kommer i diskussionskapitlet.

Bertils krav har uppnåtts förutom en. De kravet som ej uppnåddes var att PID-regulatorn skulle försetts med anti-windup och det är beroende på tidsbrist. Däremot har en PID-regulator skapats för systemet som fungerar på samma sätt som det verkliga systemet samt har manuell styrning skapats som körs via en vridpotentiometer.

6.3 Hanterande av missade och ändrade krav

Som förklarat i resultatkapitel 5 har några krav ändrats, dessutom har ett krav inte utförts. Det krav som ändrades var kravet gällande implementeringen av PLC. Vi ändrade på kravet är för att vi båda kände att det skulle vara repetitivt att ha med det då vi kan utföra samma funktionalitet med hjälp av spänningsaggregatet. PLC skulle ha möjligheten att ändra på varvtalet men varvtalet kan redan ändras via spänningsaggregatet, genom att sänka/höja spänningen kan vi få samma funktionalitet.

När det gäller att skicka en analog signal på vattennivån kände vi att det var ett lite onödigt krav då vi inte ser principen med det då ultraljudsgivaren redan skickar signalen till Arduinon för att reglera systemet. Vi diskuterade med Veronica på plats kring hur vi tänkte och hon höll med om tankesättet och därför har dessa två kraven ändrats. Anti-windupkravet har inte utförts på grund av bristande tid och prioriteringar. Anti-windup är viktigt att få med i ett system men vi prioriterade de starka kraven. Anti-windup var ett önskemål som vi tyvärr inte hann utföra för att det var en hel del komplikationer som skedde med arbetet och kan implementeras i framtiden då det inte bör vara så svårt att göra.

6.4 Framtiden

Det finns många utvecklings möjligheter för arbetet då det är så brett och fritt. I framtiden kan man överföra systemet från bredboarden till en PCB. Detta hade

gjort att systemet blev mer stilrent och att det inte är kablar överallt. Dessutom kunde vi skapa en anslutning för potentiometrarna via Blåtand eller wifi-anslutning så vi kan skicka data till Arduinon trådlöst. Systemet kunde också implementeras för en ytterligare tank såsom systemet i Lindholmen. Systemet kan skapas i en större skala genom att öka höjden till 60 cm så att det blir exakt samma storlek som det verkliga systemet. Detta hade bidragit till ännu mer precisa jämförelser. På vårt system använder vi en annan vattenpump som är satt konstant till 5 V som störning, vi kunde införa en ventil i systemet såsom det verkliga systemet för en bättre störning. Dessutom kan anti-windup implementeras för systemet.

6.5 Hållbarhet

Examensarbetet kan demonstrera fördelarna ur hållbarhetsperspektivet. Genom att utföra examensarbetet i en mindre skala än det verkliga systemet, så kan det visa på en miljömässig nytta. Detta koncept kan sedan replikeras i laboratoriemiljöer, vilket blir mer hållbart för Chalmers tekniska högskola. Dessutom är vårt system inte alltid anslutet till strömförsörjningen. Till skillnad från de system som finns på campus och alltid är anslutna till vägguttaget, vilket fortfarande förbrukar el.

Framtida applikationer för reglerteknik och Arduino-baserade vattentankar erbjuder en spännande potential för att optimera och automatisera vattenhanteringen. Genom att implementera avancerade reglertekniker och algoritmer kan vi uppnå en effektivare vattenförbrukning, bättre resursutnyttjande och minskad påverkan på miljön. Genom att fortsätta utforska och utveckla dessa applikationer kan vi forma en mer hållbar och smart framtid för vattenhantering.

7
Slutsats

Resultatet från de tester som utförts har indikerat att examensarbetet har lyckats. Det kan vi se på simulationerna som utförts i resultats kapitlet 5. Vi kan dra slutsatsen från simulationerna att det fungerar att implementera det verkliga systemet som ett proof of concept system med Arduino. Båda simuleringarna visade mycket intressanta resultat och gav indikationer på att arbetet har varit framgångsrikt. Skillnaderna i resultaten berodde på användningen av olika komponenter mellan det verkliga systemet och proof of concept systemet. Utöver det har vi också framgångsrikt implementerat proof of concept systemet med ny teknik genom användning av Arduino.

Litteraturförteckning

[1] C. -K. Park et al., Definition of common PLC MIB and design of MIB Mapper for multi-vendor PLC network management,"IEEE International Symposium on Power Line Communications and Its Applications, pp. 152-157, 2008.

[2] B. Thomas, *Modern Reglerteknik*, 5te uppl.,Göteborg:Liber AB,2016.

[3] C. Knospe,*PID control*, Mar.2006.[ONLINE].Tillgänglig:DOI:10.1109/MCS.2006.1580151, Hämtad: 2023-04-03.

[4] G. J. Silva, A. Datta, S. P. Bhattacharyya,PID Controllers for Time-Delay Systems, 1th ed.Massachusetts,2005, pp. 7-23.Accessed:05-04-2023.[Online]. Available: https://link.springer.com/book/10.1007/b138796 .

[5] Cohen, G. H., and Coon, G. A. Theoretical Consideration of Retarded Control.ÄSME. *Trans. ASME.* July 1953; 75(5): 827–834.

[6] N. Kuyvenhoven, PID tuning methods: An automatic PID tuning study with Mathcad,"in Calvin College ENGR, 315, 2002.

[7] B. Mohapatra, R, Mohapatra,J. Jijnyasa, Z. Shruti,*Easy performance based learning of arduino and sensors through Tinkercad* (2020). [ONLINE].Tillgänglig: https://cyberleninka.ru/article/n/easy-performance-based-learning-of-arduino-and-sensors-through-tinkercad/viewer Hämtad: 2023-04-06.

[8] D. M. Fars and M. B. Mahmood, *Data Acquisition of Greenhouse Using Arduino*, Journal of Babylon University/Pure and Applied Sciences, vol.22, 2014, Hämtad: 2023-04-08

[9] N. Zlatanov, *Arduino and open source computer hardware and software*, J. Water, Sanit. Hyg. Dev, vol.10, 2016, Hämtad: 2023-04-08

[10] J.Chandramohan1, R. Nagarajan, K. Satheeshkumar, N. Ajithkumar, P. A. Gopinath, S. Ranjithkumar, *Intelligent Smart Home Automation and Security System Using Arduino and Wi-fi*,International Journal Of Engineering And Computer Science , vol.6, mars 2017, Hämtad: 2023-04-09

[11] Y. Irawan, E. Sabna, A. F. Azim, R. Wahyuni, N. Belarbi, M. M. Josephine, *AUTOMATIC CHILI PLANT WATERING BASED ON INTERNET OF THINGS (IOT)*,Journal of Applied Engineering and Technological Science , vol.3, Juni 2022, Hämtad: 2023-04-09

[12] Arduino *Ärduino® MEGA 2560 Rev3,"* A000067 datasheet,Hämtad: 2023-04-03

[13] T. B. Greenslade Jr., *The potentiometer*, The Physics Teacher, vol. 43, no. 4, pp. 232-235, Jul. 2005.

[14] J. R. Kamasani,S. J. Yadala, U. S. Nakkapalli, *Arduino Based Home Electronics Labs*, Blekinges tekniska högskola, 2019, Hämtad: 2023-04-09

c

[15] K. Taneja and S. Bhatia, Äutomatic irrigation system using Arduino UNO,"2017 International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2017, pp. 132-135

[16] T. H. Nasution, M. A. Muchtar, I. Siregar, U. Andayani, E. Christian and E. P. Sinulingga, "Electrical appliances control prototype by using GSM module and Arduino,"2017 4th International Conference on Industrial Engineering and Applications (ICIEA)Accessed:05-04-2023.[Online]. Available: https://ieeexplore.ieee.org/abstract/document/7939237 .

[17] Thurlby Thandar Instruments Ltd, *"EL-R SERIES LINEAR BENCH POWER SUPPLIES,"* Thurlby Thandar Instruments Ltd, England Cambridgeshire, Datablad nummer 48511-1170, Februari 2009.

[18] .V Kh Abdrakhmanov1, R B. S.alikhov1 and S A. P.opov, *Experience of Using EasyEDA to Develop Training Boards on the PIC16f887 Microcontroller*, International Conference on Automatics and Energy ,2021, Hämtad: 2023-06-17

A

Appendix 1

```
// include the library code:
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 22, 5, 4, 3, 2);
LiquidCrystal lcd2(12, 24, 5, 4, 3, 2);

// Global Variables
int pump1 = 11;
int pid_P = 0;            // PID I Value
int pid_I = 0;            // PID I Value
int pid_D = 0;            // PID D Value
const int trigPin = 9;    //Ultrasonic Trig
const int echoPin = 10;   //Ultrasonic Echo



//SWITCH
int Auto1 = 52;
int Manual1 = 50;



//PID variables
double error = 0;
double last_error = 0;
double integral = 0;
double derivative = 0;
double output = 0;

//setpoint and input variables
double setpoint1 = 0;  //this is the desired water level in cm
double delay1 = 0;  //this is the desired water level in cm
double input = 0;

//water level sensor variables
long duration;
int distanceCm;
float procent;
// Standard Setup Function
void setup() {
  Serial.begin(9600);
  pinMode(pump1, OUTPUT);  // variant low/high
```

```
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  lcd2.begin(20, 4);
  // Clear the LCD.
  lcd.clear();
  lcd2.clear();

  pinMode(Auto1, INPUT_PULLUP);
  pinMode(Manual1, INPUT_PULLUP);
}


// Display The Values
void displayValues() {


  lcd.setCursor(0, 0);
  lcd.print("P:");
  lcd.print(pid_P);
  lcd.print("    ");

  lcd.setCursor(4, 0);
  lcd.print("I:");
  lcd.print(pid_I);
  lcd.print("    ");


  lcd.setCursor(1, 2);
  lcd.print("D:");
  lcd.print(pid_D);
  lcd.print("    ");


  lcd2.setCursor(0, 0);
  lcd2.print("Water level: ");  // Prints string "Distance" on the
     LCD
  lcd2.print(procent);     // Prints the distance value from the
     sensor
  lcd2.print("%  ");

  lcd2.setCursor(0, 1);
  lcd2.print("Setpoint: ");  // Prints string "Distance" on the LCD
  lcd2.print(setpoint1);      // Prints the distance value from the
     sensor

  lcd2.print("% ");
  lcd2.setCursor(0, 2);
  lcd2.print("Delay: ");  // Prints string "Distance" on the LCD
  lcd2.print(delay1);      // Prints the distance value from the
     sensor
  lcd2.print("s");
}

// Main Program Loop
```

```
void loop() {
  int a1 = analogRead(A1);         // Read Potientiometer value
  int a0 = analogRead(A5);         // Read Potientiometer value
  int a2 = analogRead(A2);         // Read Potientiometer value
  int d1 = analogRead(A3);  // Read Potientiometer value
  int setpoint = analogRead(A4);  // Read Potientiometer value
  int state1 = digitalRead(Auto1);
  int state2 = digitalRead(Manual1);



  // Adjust values independently

  pid_P = map(a1, 1023, 0, 0, 10);
  pid_I = map(a0, 1023, 0, 0, 10);
  pid_D = map(a2, 1023, 0, 0, 10);
  setpoint1 = map(setpoint, 1023, 0, 0, 100);
  delay1 = map(d1, 1023, 0, 0, 3);




  //calculate the input
  input = procent;

  //calculate the error
  error = setpoint1 - input;

  //calculate the integral
  integral = integral + error;

  //calculate the derivative
  derivative = error - last_error;

  //calculate the output
  output = pid_P*error + pid_I*integral + pid_D*derivative;
  //update the water pump or valve




  displayValues();

  if (state1 == 1 && state2 == 1) {
    // Switch in position 1
    // Code for action when switch is in position 1
    lcd2.setCursor(0, 3);
    lcd2.print("MODE: OFF           ");
    digitalWrite(pump1, LOW);

  } else if (state1 == 0 && state2 == 1 && output > 0) {
    // Switch in position 2
    // Code for action when switch is in position 2
    lcd2.setCursor(0, 3);
    lcd2.print("MODE: Auto           ");  // Prints string "Distance
        " on the LCD
```

```
    digitalWrite(pump1, HIGH);

  } else if (state1 == 1 && state2 == 0) {
    // Switch in position 3
    // Code for action when switch is in position 3
    lcd2.setCursor(0, 3);
    lcd2.print("MODE: Manual          ");  // Prints string "
        Distance" on the LCD
     delay(delay1*1000);
    digitalWrite(pump1, HIGH);

  } else {

    digitalWrite(pump1, LOW);
  }


  // Display the updated values

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distanceCm = 20 - (duration * 0.034 / 2);  // 20 is the height of
      where the ultrasonic is placed
  procent = (distanceCm*100)/10; // Divided by 10 because only 10
     cm of the cylinder will be used for water the remaning 10 will
      be used as a faulty precausion
  last_error = error;
  Serial.print("procent = ");
  Serial.println(procent);

  Serial.print("OUTPUT = ");
  Serial.println(output);
  Serial.println("");
  Serial.println("");

  Serial.print("Distance = ");
  Serial.println(distanceCm);

}
```

INSTITUTIONEN FÖR något ämne
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige
www.chalmers.se

CHALMERS