Automation med Arduino och Raspberry Pi

Richard Gottberg

Examensarbete för ingenjör (YH)-examen

Utbildningsprogrammet för Automationsteknik och IT

Raseborg 2016

EXAMENSARBETE

Författare:	Richard Gottberg
Utbildningsprogram och ort:	Automationsteknik och IT, Raseborg
Inriktningsalternativ/Fördjupning:	Elplanering
Handledare:	Kim Roos
Titel:	Automation med Arduino och Raspberry Pi

Datum: 25.4.2016 Sidantal: 26 Bilagor: 2

Sammanfattning

För att lära mig mera om öppenhårdvaran Arduino och Raspberry Pi så har jag utfört ett projekt där ett klot hålls balanserat på en platta med automation. I projektet har jag använt mig av en Raspberry Pi som med en webbkamera filmar var klotet befinner sig på plattan och skickar kommandon till Arduinon. Arduinon styr i sin tur två stycken servomotorer som bestämmer plattans lutning.

Examensarbetet behandlar vad Arduino automation och Raspberry Pi är och varför det är lämpligt att med hjälp av dem bekanta sig med automationens grunder. Projektet fördjupar sig i installationer och programmeringen av Arduinon och Raspberry Pin. Därtill diskuteras vilka för- och nackdelar som finns med dessa produkter.

Språk: Svenska

Nyckelord: Automation, Arduino, Raspberry Pi

BACHELOR'S THESIS

Author:	Richard Gottberg
Degree Programme:	Automation Engineering and IT, Raseborg
Specialization:	Electrical Systems Design
Supervisor:	Kim Roos
Title:	Automation Using Arduino and Raspberry Pi

Date: 25 April 2016 Number of pages: 26 Appendices: 2

Summary

To learn more about the open hardware Arduino, I have created a project where a ball is balanced on a plate with the help of automation. I have used a Raspberry Pi to control an Arduino in my project. The Raspberry Pi uses a webcam to record the position of the ball and sends commands to the Arduino. The Arduino controls two servo motors that are regulating the angle of the plate.

The purpose of this thesis is to explain what Arduino automation and Raspberry Pi are and why exploring these technologies is such a good way to get a broader understanding of automation. The project will handle in more detail how to install and program Arduino and Raspberry Pi. Furthermore, the advantages and disadvantages of these technologies are discussed.

Language: Swedish

Key words: Automation, Arduino, Raspberry Pi

OPINNÄYTETYÖ

Tekijä:	Richard Gottberg
Koulutusohjelma ja paikkakunta:	Automationsteknik och IT, Raasepori
Suuntautumisvaihtoehto/Syventävät opinnot:	Sähkösuunnittelu
Ohjaaja:	Kim Roos
Nimike:	Automaatio Arduinolla ja Raspberry Pi:llä

Päivämäärä: 25.4.2016 Sivumäärä: 26 Liitteet: 2

Tiivistelmä

Oppiakseni enemmän avoimista laitteistoista, Arduinosta ja Raspberyy Pi:stä, olen suorittanut projektin, joka käsittää pallon pysymisen tasapainossa levyllä automaation avulla. Projektissa on käytetty Raspberry Pi:tä, joka webkameran avulla kuvaa pallon sijaintia levyllä ja lähettää komennon Arduinolle. Arduino ohjaa vuorostaan kahta servomoottoria, jotka määräävät levyn kaltevuuden.

Opinnäytetyössä tarkastellaan, mitä Arduino automaatio ja Raspberry Pi ovat, ja miksi on sopivaa niiden avulla tutustua automaation perusteisiin. Projektissa käsitellään Arduinon ja Raspberry Pin asennuksia ja ohjelmointia. Sen lisäksi pohditaan, mitkä ovat näiden tuotteiden hyvät ja huonot puolet.

Kieli: Ruotsi

Avainsanat: Automaatio, Arduino, Raspberry Pi

Innehållsförteckning

1 Inledning

Detta examensarbete går igenom hur man med enkla och förmånliga medel kan tillverka en apparat som med hjälp av automation som håller ett klot balanserat på en platta. Målet är att apparaten skall kunna fungera helt automatiskt utan utomstående påverkan. För det här projektet så valdes Arduino och Raspberry Pi för att hålla kostnaderna så låga som möjligt. Arduino kommer att sköta servomotorerna som reglerar plattans lutning. Raspberry Pin kommer att fungera som hjärna som med hjälp av en webbkamera ser var klotet befinner sig på plattan. Raspberry Pin kommer att räkna ut vilken lutning plattan kommer att behöva och skicka ut kommandon till Arduinon som sedan reglerar plattans vinkel.

Examensarbetet delas upp i flera delar. Först beskrivs vad en enkortsdator Raspberry Pi är och hur man kommer igång med att använda en Raspberry Pi. Examensarbetet fortsätter sedan med att beskriva vad Arduino är och vad man kan använda Arduino till. De komponenter som går att koppla ihop med Arduino beskrivs, och hur man programmerar Arduino och får den att fungera ihop med kringutrustning. Därtill behandlas varför så många väljer att använda Arduino och Raspberry Pi.

Slutligen behandlas själva projektet, det vill säga hur apparaten är uppbyggd (vilka komponenter som finns var och hur de fungerar ihop). Mjukvaran är det viktigaste i projektet och det är mjukvaran som styr hårdvaran och får komponenterna att utföra de uppgifter man vill att de skall utföra. Arbetet beskriver vilka mjukvaruprogram som behövs och hur man programmerar de komponenter som apparaten använder.

2 Bakgrund

2.1 Raspberry Pi

Raspberry Pi är en enkortsdator med en PC:s viktigaste komponenter i mycket kompakt format på ett litet kretskort (Figur 1). Man behöver endast ett SD kort där man kan installera ett operativsystem. Datorn Raspberry Pi utvecklades för att göra det enklare och förmånligare för skolor att undervisa grunderna i datoranvändning och -programmering. Datorn utvecklades i England av Raspberry Pi Foundation. De första versionerna av Raspberry Pi kom ut 2012, då Raspberry Pi modell A och modell B introducerades. (Wikipedia, 2016a)

Raspberry Pi fungerar med 5V spänning som den får via en micro USB kabel. Det vanligaste operativsystemet som Raspberry Pi använder är Raspbian som är en optimerad variant av Linux Debian. Man kan också köra andra versioner av Linux på Raspberry Pi. Det kommer ut nya versioner av Raspberry Pi som är effektivare i takt med utvecklingen av nya mikrochips. Idag finns det Raspberry Pi Modell A, Raspberry Pi Modell A+, Raspberry Pi Modell B, Raspberry Pi modell B+, Raspberry Pi 2 Modell B, Raspberry Pi 3 Modell B och Raspberry Pi Zero. Modell A saknar Ethernet förbindelse medan modell B har Ethernet förbindelse. Alla modeller har en ARM processor vilket betyder att operativsystemet måste vara kompilerat för ARM. Det har inte funnits stöd för andra operativsystem än Linux före Raspberry Pi 3. Nu har Microsoft lovat att de skall ge ut en version av Windows 10 som fungerar på Raspberry Pi 3. (Wikipedia, 2016b)

Raspberry Pi har också en så kallad GIPO (general purpose input-output connector) som gör det enklare att koppla och programmera extern utrustning till den. Man kan använda GIPO pinnarna till att koppla sensorer som skickar data till Raspberry Pin eller så kan man koppla t.ex. LEDar eller brytare som styrs av Raspberry Pin. (Wikipedia, 2016b)

Figur 1. Bild på Raspberry Pi model B (Tomre, 2016).

2.2 Arduino

Arduino är skapad med öppen hårdvara med syftet att göra det enkelt, tillgängligt och förmånglit att lära sig grunderna i automation och programmering. Arduino består av hårdvara i form av ett mikrokontrollerkort och mjukvara som installeras på en dator där man programmerar Arduinon. (Arduino, 2016)

Det finns flera typer av mikrokontrollerkort som klarar av olika sorters uppgifter, som att styra elektronisk utrustning och ta emot data från sensorer. Mikrokontrollerkortet är skapat med en enkel och öppen kretslösning, och har både in- och utgångar. Man kan dessutom koppla så kallade sköldar till mikrokontrollerkortet som gör att man får flera in- och utgångar. (Arduino, 2016)

Mjukvaran är en multiplattformsapplikation, d.v.s. mjukvaran fungerar på flera olika plattformar som Windows, Mac och Linux. Mjukvaran för den är därför skriven i Java. Det är enkelt att installera Arduinos mjukvara (IDE), den kan laddas ner från Arduinos officiella

webbsida och installeras på datorn. Arduino IDE är väldigt enkel att komma igång med och det finns färdigt installerade program som man kan prova utan att själv behöva programmera. När man har IDE installerat, så kan datorn kopplas ihop med hårdvaran och sedan kan programmen till Arduino mikrokontrollerkortet laddas upp. (Hem automation för alla, 2016)

2.2.1 Historia

År 2005 skapades det första Arduino kortet av Massimo Banzi på Ivreas institutionen för interaktiv design, Italien. Målsättningen för Arduino projektet var att skapa en plattform för studerande att bekanta sig med automation. Kriterierna var att det skulle vara förmånligt att tillverka korten och enkelt att programmera dem. När man skapade den första Arduinon så använde man sig av en ATmega integrerande krets. Denna grundande sig på platformen Wiring, som den columbianska studenten Hernando Barragan skapade år 2004. Wiring består av hårdvara byggd kring en ATmega integrerad krets och en IDE som är baserad på Processing som gör det enkelt att programmera, men Adruino projektet förbättrade mjukvaran så att den skulle bli ännu mer användarvänlig. Arduino är en öppen hårdvara vilket betyder att vem som helst får använda och vidare utveckla produkter med Arduino utan att behöva betala licensavgifter. (Circuits Today, 2016)

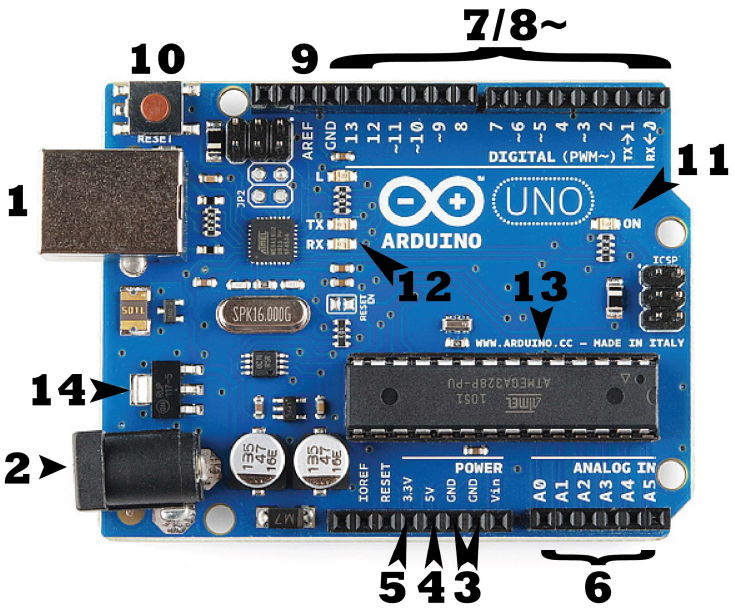
2.2.2 Hårdvaran

Arduino är en mycket enkel typ av dator som inte har något eget operativsystem. Man behöver därför en PC med en Arduino IDE som består av en kompilator och en bootloader. Arduinon har ett väldigt litet minne, till exempel, Arduino Uno har 32 kB minne. Det är ändå tillräckligt för de flesta program, eftersom Arduinons uppgift är att med enkla kommandon styra kringutrustning och ta emot data från sensorer. Arduino kan direkt styra kringutrustning

med hjälp av data som den samlar in, eller skicka data till en PC som behandlar data och därefter skickar kommandon till Arduinon. (Kjell&Company, 2016)

2.2.3 Arduino kortet

Det finns många olika typer av Arduino mikrokontrollerkort och ett av de vanligaste korten är Arduino Uno (Figur 2). Arduino Uno är ett bra mikrokontrollerkort när man vill lära känna Arduino, eftersom det har allt som behövs för att utföra enkla projekt. Arduino Uno har 14 digitala in/utgångar, 6 stycken analoga ingångar, USB kontakt, elintag och en återställnings knapp. (Kjell&Company, 2016)



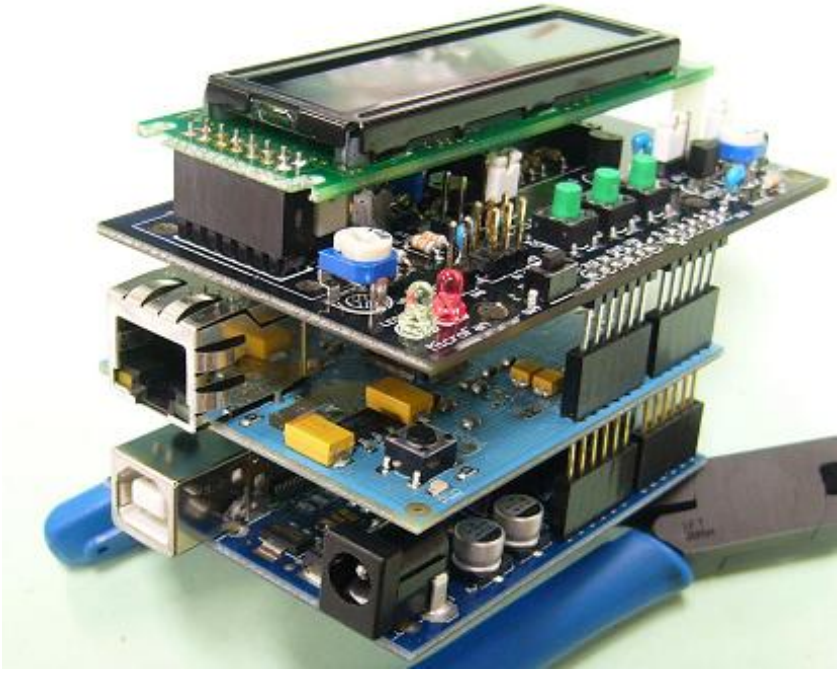
Figur 2. Arduino Uno. 1: USB kontakt, 2: strömingång, 3-9: kopplingspinnar, 10: återställningsknapp, 11: på/av, 12: TX och RX led, 13: integrerad krets, 14: spänningsreglerare (Sparkfun, 2016).

För att fungera så behöver Arduino Uno en strömkälla. Det går att förse Arduino Uno med ström via en USB kontakten (Figur 2; 1) eller strömingången (Figur 2; 2). USB kontakten används också när man överför program från en dator till Arduino. Det rekommenderas att man använder mellan 6 och 12 Volt spänning till Arduino och helst inte över 20 Volt. Kopplingspinnarna på Arduino Uno kortet är numrerade 3-9 (Figur 2) och det är viktigt att de följer en viss standard som är densamma för alla Arduino kort. Det gör det möjligt för vem som helst att tillverka moduler och sköldar som passar till alla Arduino kort och det underlättar sammankoppling av dessa enheter. I Figur 2 pekar nummer 3 mot jordpinnarna, det kan finnas flera kopplingspinnar till jorden. Nummer 4 pekar mot 5V potential pinnen, 5 pekar mot 3V potential pinnen, 6 pekar mot de analoga pinnarna som är numrerade A0-A5. Nummer 7/8 pekar mot de digitala pinnarna. Där är 3,5,6,9,10,11 så kallade PWM (pulsbredsmodulerings) pinnar och har ett (~) tecken framför sig. Det betyder att de är digitala pinnar som kan efterlikna analoga pinnar. Nummer 9 pekar mot analog referens (AREF) som sällan används, men om man har analog utrustning som använder sig av ett smalare spänningsområde än 0V-5V så går det att ställa in den övre gränsen för spänningsorådet (t.ex. 0V-2V). Genom att trycka på återställningsknappen (10) så upprepar man de program som finns uppladdade på en Arduino. På/av ledd (11) visar om Arduino är kopplad till en strömkälla, medan TX och RX ledderna (12) visar om Arduinon tar emot eller skickar ut data (t.ex. om man laddar upp ett program till en Arduino). Integrerad krets (13) är kretsen där alla beräkningar i en Arduino utförs. Olika Arduino kan ha olika kretsar men de flesta använder ATMEL företagets ATmega seriekretsar. Spänningsregleraren (14) reglerar strömmen som en Arduino tar emot. Den klarar av att reglera en spänning upp till 20 volt. (Sparkfun, 2016)

2.2.4 Kringutrustning

En Arduino är skapad till att styra elektronisk utrustning och ta emot data från sensorer. Man kan koppla sensorer och annan utrustning direkt till Arduinon eller så kan man använda sig av så kallade sköldar. Sköldarna ger Arduinon extra egenskaper som t.ex. möjlighet att koppla

upp den till internet. Sköldarna är ofta anpassade i storlek så att de passar direkt på mikrokontrollerkortet (ofta Arduino Uno). Sköldarna är dessutom planerade så att de pinnar som inte används går direkt genom skölden och kan därför användas av andra sköldar. Man kan också placera flera sköldar på varandra. Detta leder till att man kan få många tilläggsfunktioner kompakt och lätt till en Arduino. En del sköldar använder sig av samma pinnar som andra, vilket gör att alla sköldar inte är kompatibla med varandra. (Kjell&Company, 2016)



Figur 3. Arduino Uno med två stycken sköldar; en Ethernet sköld och en annan sköld som har en display och några andra funktioner. (Microfan, 2016)

Moduler består av flera komponenter som är färdigt tillverkade för ett visst ändamål. Till exempel Wifi modulen består av de komponenter som behövs som sedan är färdigt ihop lödda. Modulerna kopplas oftast inte direkt till mikrokontrollerkortet eller skölden, utan man använder sig av kablar för att koppla in modulen. Det betyder att man kan välja vilka pinnar man vill när man kopplar in modulen. Modulerna kan användas till andra ändamål än Arduino. (Kjell&Company, 2016)

2.3 Mjukvara

Med hjälp av mjukvara kan man styra programmerbar hårdvara. Arduino och Raspberry Pi är bägge programmerbar hårdvara. Man kan inte skriva program direkt till Arduino så det behövs en PC som kan ladda upp program till Arduinon. Raspberry Pi är en PC som man kan använda till att skriva, kompilera och ladda upp program till Arduinon.

2.3.1 Arduino IDE

Arduino IDE (integrerad utvecklingsmiljö) är ett mjukvaruprogram som installeras på en PC. Man kan skriva, kompilera och ladda upp program till Arduino med Arduino IDE. Arduinos IDE är skrivet i Java vilket gör det enkelt att få den kompilerad till olika plattformer som Windows, Mac och Linux. Arduinos IDE är skapat att vara mycket enkelt att använda och man programmerar programmen i C/C++ liknande kod men man behöver endast använda sig av två funktioner. Dessa är ”Setup()” som finns i början på programmet och körs bara en gång där man b.la. kan definiera variabler som programmet använder, och ”Loop()” som upprepar de kommandona som finns inom loopen så länge Arduinon är påslagen. (Hem automation för alla, 2016)

2.3.2 Processing

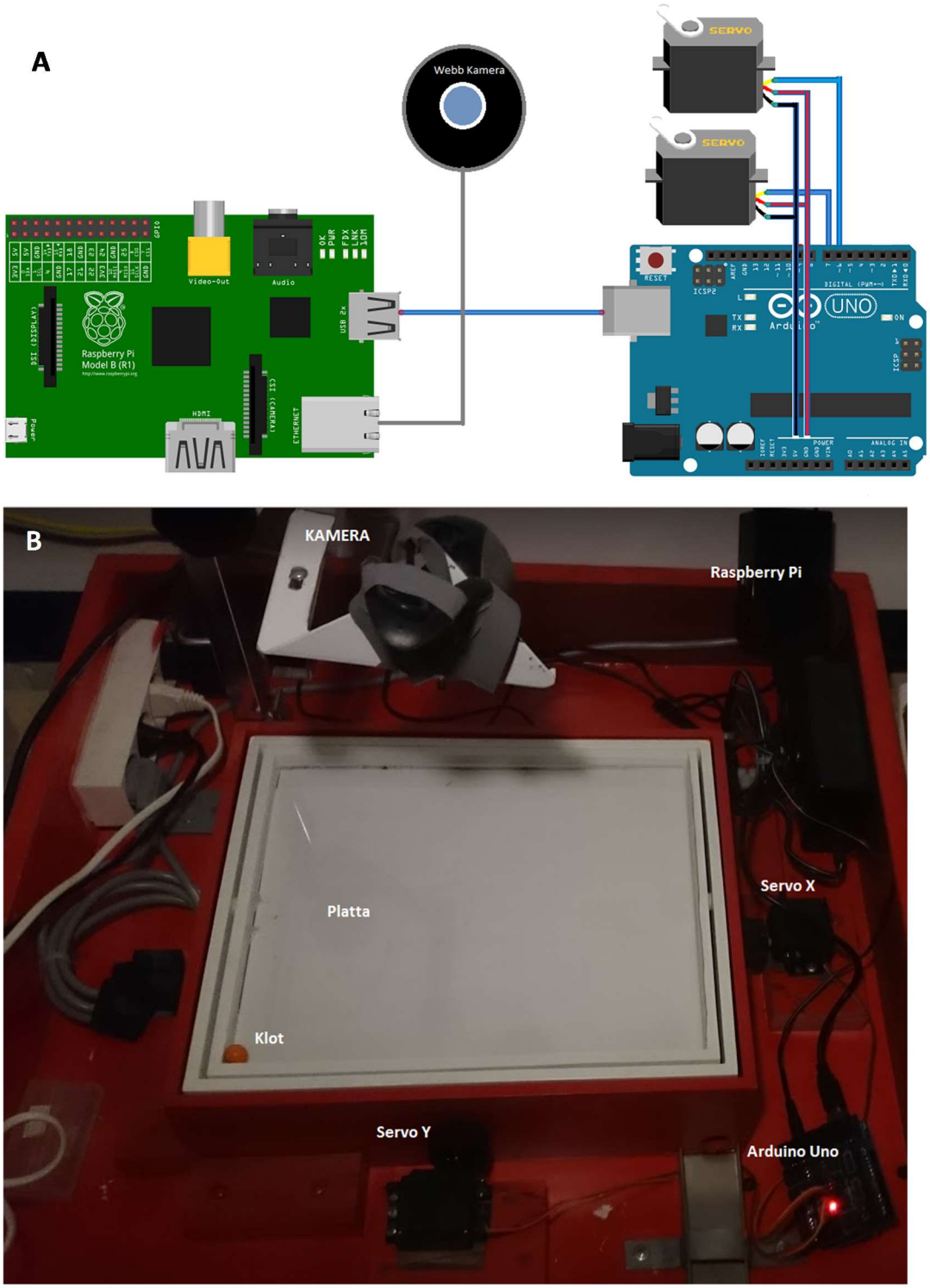
För det här projektet valdes Processing som programmeringsspråk för programmeringen av Raspberry Pin. För liknande uppgifter har ofta C++ eller Processing använts. C++ är mycket intressant och det har utvecklats ett programvarubibliotek till C++ som heter OpenCV som gör det enklare att göra bl.a. bildanalyser. Med OpenCV kan man analysera en webbkameras bilder och t.ex. följa med ett objekt på video, vilket är en viktig del av detta projekt. (Ubuntu documentation, 2016)

Processing är ett ganska nytt programmeringsspråk som är skapat för att programmera elektronisk konst, och det har också en stark anknytning till Arduino. Processing har ett video programvarubibliotek som underlättar bildanalyser. Med Processings videobibliotek kan man analysera bilder från webbkameran att följa med ett objekt. Processing har ett användarvänligt IDE som finns tillgängligt till både Raspberian och Windows. Processing baserar sig på öppen källkod. Det är skapat från programmeringsspråket Java, men använder sig av en förenklad form av grafikprogrammering. Processing valdes för detta projekt eftersom det är ett nytt och intressant språk, som borde ha alla förutsättningar att kunna klara av att förverkliga projektet. (Processing, 2016)

3. Metoder och resultat

Som projekt till mitt slutarbete valde jag att med automation balansera ett klot i mitten på en platta. Under en tidigare kurs hade en grupp studerande utfört automationsprojektet med en annan dator än Raspberry Pi, men fick inte klotet att hållas på mitten av plattan. Jag fick anordningen som var färdigt byggd samt alla komponenter till projektet från min handledare (Figur 4). Därför kunde jag koncentrera mig på att koppla ihop delarna och få mjukvaran att fungera.

Projektet delades in i två huvudsakliga delar; (1) koppling av komponenter och installation av program, samt (2) programmering av Raspberry Pi och Arduino. Arduino Uno kontrolleras av Raspberry Pi som med hjälp av en webbkamera kontrollerar var klotet finns på plattan. För att kunna börja programmera Arduinon så krävdes det att Raspberry Pin hade alla program och webbkameran installerade. När Raspberry Pi hade ett operativsystem, webbkamera, Arduino IDE och ett programmeringsspråk installerat så kunde jag för första gången koppla upp mig till Arduino Uno med hjälp av Arduino IDE.



Figur 4. (A) Kopplingsschema och (B) anordningen som balanserar klotet på plattan. Den består av två servomotorer (x, y), Arduino Uno, Raspberry Pi, kamera samt labyrintspel.

3.1 Installationer

Först installerade jag ett operativsystem till Raspberry Pi. Jag valde att installera Raspbian eftersom det är det vanligaste och mest utvecklade operativsystemet för Raspberry Pi, och för att det är lätt att hitta information om man behöver hjälp med något när man använder det operativsystemet. Att installera Raspbian är enkelt eftersom det är skräddarsytt för Raspberry Pi. Man behöver endast ställa in vilket tangentbord och vilket språk man vill använda.

Det är okomplicerat att installera de program som har fullt stöd för Linux. Det enklaste sättet att installera program på en Linux maskin är att använda sig av terminalen och skriva in installationskommandon. Om det inte har fullt stöd så måste man ladda ned programmet och installera allt manuellt. Att installera manuellt är mycket besvärligare, det finns en stor risk att något går fel på vägen och att programmet inte fungerar som planerat.

Det första programmet jag installerade var Arduino IDE. Arduino IDE har fullt stöd för Linux och det installerades automatiskt när jag skrev in installationskommandon i terminalen. Sedan installerade jag Processing version 2, vilket var mera komplicerat eftersom det inte fanns ett installationskommando tillgängligt. Det fanns ändå bra beskrivningar på webben över hur man skulle gå tillväga (Källa 10). Processing version 3 och nyare versioner har automatiska installationskommandon så numera är det lättare att installera Processing till Raspbian. För att kunna programmera Arduino med Processing så behöver man också installera Arduino biblioteken till Processing. Instruktioner över hur man går tillväga finns tillgängliga på webben. (Scruss, 2016)

3.2 Programmering

När alla installationer var gjorda prövade jag om det gick att få kontakt med Arduinon från Raspberry Pi med Arduino IDE. För detta finns olika testprogram färdigt installerade, som man kan ladda upp till Arduinon. Jag använde mig av ett testprogram som heter Blink. Blink får en led på Arduinon att börja blinka och då vet man att det fungerar. När jag hade skapat kontakt med Arduinon prövade jag om det med hjälp av Processing gick att få kontakt till och kontrollera Arduinon. Till detta behövdes ett program skrivet på Arduino IDE som gör det möjligt för Arduino att ta emot kommandon från Processing. Det fanns ett färdigt program i Processing som gjorde att samma led började blinka. När programmeringsprogrammen var installerade och kontakterna fungerade kunde jag börja programmera regleringen av servomotorerna och analysering av webbkameran.

3.2.1 Programmera Arduino

För att Arduinon ska utföra kommandon behöver den programmeras. Arduinon programmeras i Arduino IDE med C/C++ som programmerings-språk. Det första program som jag laddade upp till Arduinon hette Servoprogram (Figur 5), vilket gör att en servomotor först kör åt ett håll och sedan tillbaka. Servoprogrammet gör att Arduinon självständigt kan kontrollera servomotorn. Men, med Servoprogram kan man inte styra Arduinon eller servomotorerna från en PC, för detta krävs ett annat program. Det finns färdigt ett bibliotek, Firmata, som gör att det är enklare att programmera Arduinon till att bli kontrollerad av en PC. Jag använde mig av ett program, StandardFirmata, som finns tillgängligt i Arduino IDE. Man kan använda sig av ServoFirmata om man bara vill styra två servomotorer. Dessutom är det också rätt okomplicerat att själv skriva ett program som kontrollerar servomotorerna, eftersom det finns mycket material om hur man gör det tillgängligt på webben.

Figur 5. Servoprogram i Arduino IDE.

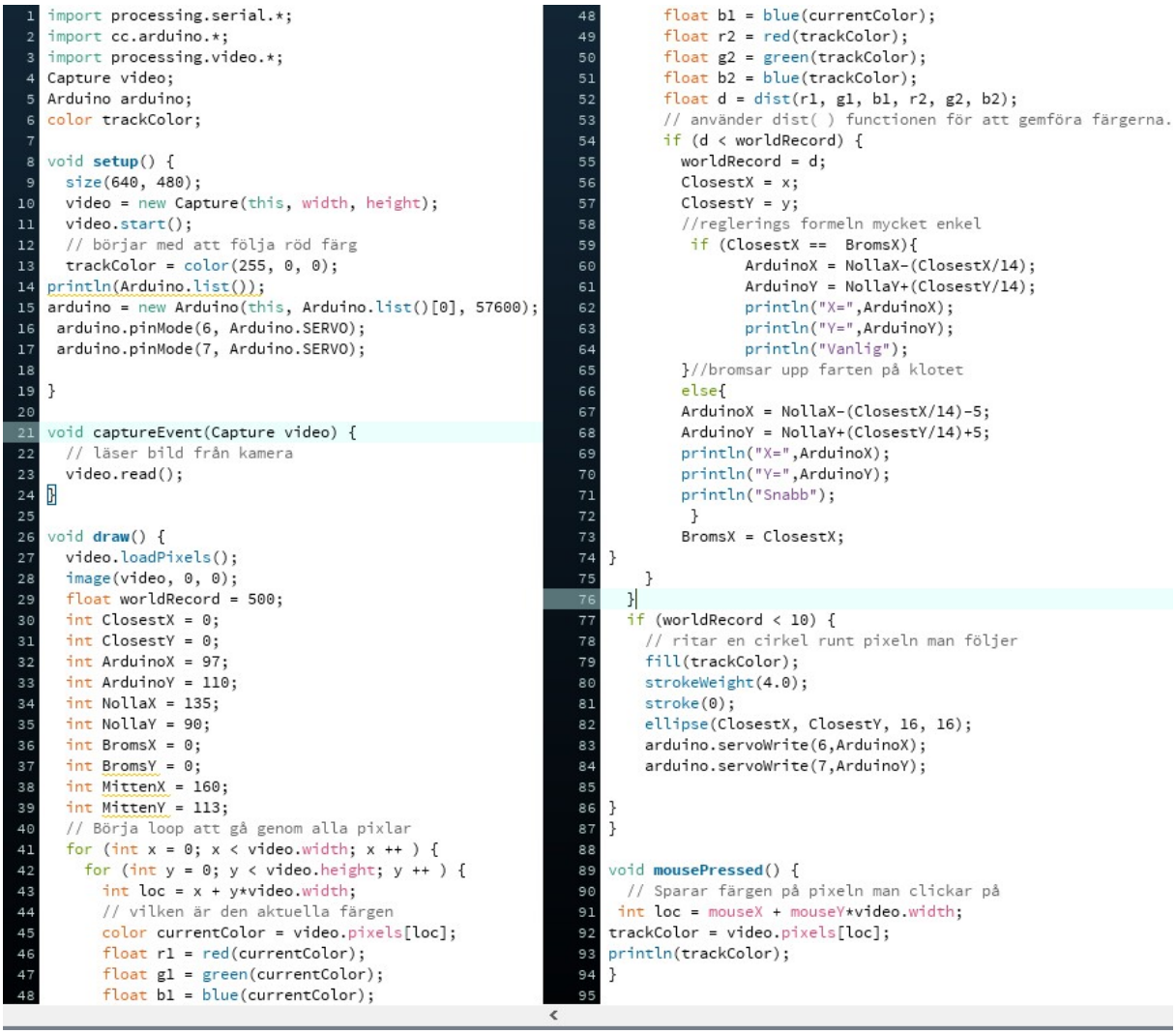
3.2.2 Programmera i Processing

Processing liknar programmeringsspråket Java, men det förenklar vissa saker som kan vara svåra för ovana programmerare. Jag har använt programmeringsspråket Java tidigare så det gjorde det lättare att förstå hur Processing fungerar. På Processings officiella webbsida finns dessutom videor och exempel på hur man kommer igång med att programmera i Processing. Jag kände inte till programvarubiblioteken Arduino och Video i Processing. Eftersom de var

viktiga för att kunna genomföra projektet läste jag hur de fungerade och prövade mig fram med hjälp av olika exempel.

I början av projektet så utförde jag all programmering på Raspberry Pi, men konstaterade snart att det inte var ett bra arbetsredskap eftersom den har en så svag processor. Att jobba med en dator med en svag processor är mycket tidskrävande, och varje gång man gör något så måste man vänta på att Raspberry Pin bearbetar data. Därför installerade jag alla program till min bärbara PC, där jag snabbare kunde kontrollera om det jag hade programmerat verkligen fungerade. På den bärbara PCn gick det lätt att pröva hur olika justeringar påverkade programmet. Min första version av programmet som balanser klotet på plattan bestod av delar från flera olika program som fanns på nätet (Figur 6). En del av programmet följde med var klotet fanns på plattan. En annan del skickade över information till Arduinon och själv skrev jag ett program som skötte regleringen av servomotorerna. Efter lite justerande av regleringsdelen i programmet fick jag klotet att hållas på plattan. Därefter överförde jag programmet till Raspberry Pin.

När jag hade fått programmet överfört och kompillerat på Raspberry Pin så konstaterade jag att det inte fungerade lika bra på den som det hade fungerat på min bärbara PC. Programmet slutade att fungera efter en kort stund och det fick aldrig klotet att hållas i mitten på plattan. Jag försökte optimera programmet genom att plocka bort alla onödiga kommandon. Till exempel tog jag bort den delen av programmet som ritar ut bilden från kameran. Åtgärderna var inte tillräckliga, utan programmet slutade fortsättningsvis att fungera.



```
import processing.serial.*;
import cc.arduino.*;
import processing.video.*;
Capture video;
Arduino arduino;
color trackColor;

void setup() {
  size(640, 480);
  video = new Capture(this, width, height);
  video.start();
  // börjar med att följa röd färg
  trackColor = color(255, 0, 0);
println(Arduino.list());
arduino = new Arduino(this, Arduino.list()[0], 57600);
 arduino.pinMode(6, Arduino.SERVO);
 arduino.pinMode(7, Arduino.SERVO);

}

void captureEvent(Capture video) {
  // läser bild från kamera
  video.read();
}

void draw() {
  video.loadPixels();
  image(video, 0, 0);
  float worldRecord = 500;
  int ClosestX = 0;
  int ClosestY = 0;
  int ArduinoX = 97;
  int ArduinoY = 110;
  int NollaX = 135;
  int NollaY = 90;
  int BromsX = 0;
  int BromsY = 0;
  int MittenX = 160;
  int MittenY = 113;
  // Börja loop att gå genom alla pixlar
  for (int x = 0; x < video.width; x ++ ) {
    for (int y = 0; y < video.height; y ++ ) {
      int loc = x + y*video.width;
      // vilken är den aktuella färgen
      color currentColor = video.pixels[loc];
      float r1 = red(currentColor);
      float g1 = green(currentColor);
      float b1 = blue(currentColor);
      float b1 = blue(currentColor);
      float r2 = red(trackColor);
      float g2 = green(trackColor);
      float b2 = blue(trackColor);
      float d = dist(r1, g1, b1, r2, g2, b2);
      // använder dist( ) functionen för att gemföra färgerna.
      if (d < worldRecord) {
        worldRecord = d;
        ClosestX = x;
        ClosestY = y;
        //reglerings formeln mycket enkel
         if (ClosestX ==  BromsX){
               ArduinoX = NollaX-(ClosestX/14);
               ArduinoY = NollaY+(ClosestY/14);
               println("X=",ArduinoX);
               println("Y=",ArduinoY);
               println("Vanlig");
        }//bromsar upp farten på klotet
        else{
        ArduinoX = NollaX-(ClosestX/14)-5;
        ArduinoY = NollaY+(ClosestY/14)+5;
        println("X=",ArduinoX);
        println("Y=",ArduinoY);
        println("Snabb");
         }
        BromsX = ClosestX;
}
    }
  }
  if (worldRecord < 10) {
    // ritar en cirkel runt pixeln man följer
    fill(trackColor);
    strokeWeight(4.0);
    stroke(0);
    ellipse(ClosestX, ClosestY, 16, 16);
    arduino.servoWrite(6,ArduinoX);
    arduino.servoWrite(7,ArduinoY);

}
}

void mousePressed() {
  // Sparar färgen på pixeln man clickar på
 int loc = mouseX + mouseY*video.width;
trackColor = video.pixels[loc];
println(trackColor);
}
```

Figur 6. Första versionen av ett program som balanserar ett klot på platta skrivet i Processing. Programmet är inte dessmera optimerat.

3.3 Reglering av plattans lutning

För att få klotet att balansera på plattan så måste man kontrollera plattans lutning. Ökar man lutningen på en sida så rullar klotet till motsvarande sida. För att få klotet att hållas på mitten krävs det att plattan är vågrät och att klotet inte rör på sig. Om klotet rör på sig måste man få det att stanna. När jag prövade olika formler för att reglera plattans lutning så gjorde jag det först i ett tvådimensionellt plan, alltså först bara på x- axeln. Min första formel höjde bara

upp plattan på den sidan klotet befann sig på. Om klotet var längst ut på plattan så höjde jag upp den sidan maximalt och om klotet var nära mitten så var plattan nästan vågrät. Med denna formel så uppstod det oscillation med klotet. Klotet stannade aldrig i mitten utan åkte från sida till sida. För att klotet inte skulle få upp en så hög hastighet så minskade jag på den maximala lutningen av plattan vilket resulterade i att klotet inte mera tog i kanterna på plattan. Klotet stannade ändå inte upp i mitten utan oscillationen fortsatte. Jag lade en del till min kod som bromsade upp klotet om det rörde sig för snabbt och det visade sig vara mycket effektivt för då fick jag klotet att stanna på mitten. När jag fick klotet att stanna på x-axeln så var det bara att använda samma formel för y-axeln och då kunde klotet stanna i mitten på plattan.

Eftersom den första versionen av programmet som balanserade klotet på plattan inte fungerade i Raspberry Pi, började jag från början och använde PID reglering. Med en PID regulator kan man minska på oscillation och få en stabil nivå på det man vill kontrollera. PID regulatorn har tre faktorer för att påverka hur snabbt och effektivt något kan regleras. P-verkan eller K_r påverkar hur snabb återkopplingen är, det vill säga ett högt värde på K_r motsvarar en stor proportionell återkoppling. I-verkan eller T_i eliminerar stationära störningar, d.v.s. ett litet värde på T_i leder till stor integrerande verkan och stort T_i minskar inverkan. Om T_i är oändligt stor så försummas inverkan. D-verkan eller T_d är deriverande verkan, d.v.s. den ger bättre stabilitetsmarginaler men ökar mätfelens inverkan. Man kan skriva PID formeln enligt följande:

$$u(t) = K_r\,(e(t) + \frac{1}{T_i}\int_0^t e(t)\,dt + T_d\,\frac{de}{dt})$$

(CTMS, 2016)

3.4 Minska belastningen för Raspberry Pi

Efter mycket funderande över hur man kunde bättre optimera programmen så kom jag till lösningen att flytta över regleringsberäkningarna till Arduinon. Hittills hade Arduinon bara styrt servomotorerna medan Raspberry Pin skötte filmandet och beräknade var klotet befann sig, samt räknade ut vilken position motorerna skall ha, det vill säga regleringen. Det finns ett färdigt programvarubibliotek i C++ som heter PID_v1 som gör det enklare att programmera PID reglering för Arduino. Jag installerade PID_v1 programvarubiblioteket till Arduino IDE och började programmera ett program som skulle klara av att sköta regleringen av servomotorerna direkt på Arduino kretsen. Jag sammanställde ett enkelt program som klarade av att sköta regleringen, men det var besvärligt att få det rätt inställt. Eftersom jag hade en version som fungerade på min bärbara PC så tänkte jag att det kan vara ett passligt projekt för någon annan att fortsätta med och beslöt därför att inte fortsätta programmera på Arduinon.

4. Diskussion

4.1 Vad har jag lärt mig av projektet

Största delen av projektet har handlat om programmering och att få komponenterna att fungera med varandra. Under projektets gång har jag fördjupat mina kunskaper i hur Arduino och Raspberry Pi fungerar och det har uppstått många svårlösta och oförväntade problem. Att få alla program installerade var relativt okomplicerat men när jag skulle exekvera programmen som jag hade programmerat så uppstod det svårigheter. Speciellt mycket besvär orsakades av Raspberry Pin, på grund av att det saknades vissa drivrutiner eller för att inställningar var fel. Det var väldigt tidskrävande att reda ut alla problem eftersom när man hade löst ett problem så uppstod nästa. Under projektets gång har jag blivit bättre och effektivare på att identifiera och tackla tekniska svårigheter.

Trots att jag hade bekantat mig med både Raspberry Pi och Arduino i tidigare projekt så har jag fått betydligt djupare kunskaper om dessa teknologier genom detta projekt. Tack vare Arduino har mina kunskaper inom automation breddats och gett mig möjlighet att i framtiden utveckla egna automationslösningar.

4.2 Var kan tekniken användas?

Det är mycket aktuellt med automation och PID reglering idag. Samma tekniker används exempelvis i drönare (obemannade flygande farkoster) och ståhjulingar (ett tvåhjuligt självbalanserande fordon; t.ex. Segway). Det finns många flera liknande uppfinningar som använder sig av dessa teknologier. Inom industrin har man redan länge använt sig av regulatorer och automation, men det har ökat explosionsartat för vanliga konsumenter under de senaste åren på grund av att priserna har sjunkit till en förmånligare nivå. Arduino och Raspberry Pi kommer att ha stor betydelse för utvecklingen inom IOT (internet of things). Detta beror delvis på att det är relativt enkelt att bekanta sig med automation via dessa teknologier, och delvis på att kostnaderna för att utveckla nya smarta verktyg sjunker.

Figur 7. Drönare utgör exempel på teknik som använder sig av en PID regulator för att stabilt hållas i luften. (Illustrerad vetenskap, 2016)

4.3 Slutsats och kommentarer

Arduino och Rasperry Pi är relativt nya projekt, men när man ser på webben hur mycket information det finns om projekten så får man en klar uppfattning om deras popularitet. Både Arduino och Raspberry Pi har en ljus framtid och deras utveckling har gått väldigt snabbt framåt under den tid jag har använt mig av deras teknologier. Jag tycker att Ardunio projektet är bättre utvecklat och jag stötte på mindre problem med det. De svårigheter jag hade med Arduino berodde främst på felprogrammering eller felkoppling. Det är imponerande vad man kan åstadkomma med Arduino och jag kommer att ha mycket nytta av det jag har lärt mig om Arduino i framtiden.

Raspberry Pi har vållat mycket problem i mitt projekt. Det är enkelt att installera program till Raspberry Pi som de flesta användare använder. Att installera ovanligare program som färre använder sig av visade sig vara riktigt knepigt. Eftersom Raspbian är en nedbantad version av Debian så installeras inte allt automatiskt, utan man måste installera flera komponenter i efterhand. Vid flera tillfällen under programmerandet uppstod fel p.g.a. att det saknades vissa drivrutiner i Raspbian. Det var mycket tidskrävande att hitta lösning på problemen och det hindrade mig från att komma vidare på projektet. De flesta problemen gick att lösa genom att söka lösningar från webben, eftersom många andra hade haft liknande problem.

Referenser

Arduino, 2016. *What is Arduino.* [Online]

https://www.arduino.cc/en/Guide/Introduction. [Hämtat 9.4.2016].

Circuits Today, 2016. *Story and History of Development of Arduino.* [Online]

http://www.circuitstoday.com/story-and-history-of-development-of-arduino [Hämtat 9.4.2016].

CTMS, 2016. *Introduction to PID design*. [Online]

http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID

[Hämtat 9.4.2016].

Hem automation för alla, 2016 Introduktion till Arduino [Online]

http://blog.m.nu/introduktion-till-arduino [Hämtat 9.4.2016]

Kjell&Company, 2016. *Kom igång med Arduino*. [Online]

http://www.kjell.com/se/fraga-kjell/teman/kom-igang-med-arduino [Hämtat 9.4.2016].

Processing, 2016. *Overview.* [Online]

https://processing.org/overview/ [Hämtat 9.4.2016].

Rasbian,2016. *Welcome to Raspbian.* [Online]

https://www.raspbian.org. [Hämtat 15.4.2016].

Scruss, 2016. *Processing 2.1 + Oracle java + Raspberry Pi + serial + arduino =* ☺. [Online]

http://scruss.com/blog/2014/01/07/processing-2-1-oracle-java-raspberry-pi-serial-arduino-%E2%98%BA/ [Hämtat 9.4.2016].

Sparkfun, 2016. *What is an Arduino.* [Online]

https://learn.sparkfun.com/tutorials/what-is-an-arduino [Hämtat 9.4.2016].

Ubuntu documentation, 2016*. Open CV.* [Online]

https://help.ubuntu.com/community/OpenCV [Hämtat 9.4.2016].

Wikipedia, 2016a. *Raspberry Pi.* [Online]

https://sv.wikipedia.org/wiki/Raspberry_Pi [Hämtat 15.4.2016].

Wikipedia, 2016b. *Raspberry Pi.* [Online]

https://en.wikipedia.org/wiki/Raspberry_Pi [Hämtat 15.4.2016].

Figurer

Figur 1. Tomre, 2016. *Raspberry Pi model B*. [Online]

http://tomre.es/img/raspberrypi.jpg [Hämtat 9.4.2016].

Figur 2. Sparkfun, 2016. *Bild på Arduino.* [Online]

https://cdn.sparkfun.com/assets/b/f/e/9/c/513824face395f6d3d000000.png [Hämtat 9.4.2016].

Figur 3. Microfan, 2016. *Bild på Arduino med Sköldar*. [Online] www.microfan.jp/images/opt/clcd-booster/CLCD-BOOSTER-P4.JPG [Hämtat 9.4.2016].

Figur 4a. Kopplingsschema.

Figur 4b. Foto av anordningen som balanserar klotet på plattan.

Figur 5. Servoprogram i Arduino IDE.

Figur 6. Första versionen av ett program som balanserar ett klot på platta skrivet i Processing.

Figur 7. Illustrerad vetenskap, 2016. *Bild på Drönare.* [Online] http://illvid.dk/teknologi/droner/droner-kan-forhindre-fremtidige-ebola-udbrud [Hämtat 9.4.2016].

Bilaga 1 Processing kod

```
import processing.serial.*;
import cc.arduino.*;
import processing.video.*;

// definierar variabel Capture, Video trackColor
Capture video;
Arduino arduino;
color trackColor;


void setup() {
  size(640, 480);
  video = new Capture(this, width, height);
  video.start();
  // börjar med att följa röd färg
  trackColor = color(255, 0, 0);


arduino = new Arduino(this, Arduino.list()[0], 57600);
 arduino.pinMode(6, Arduino.SERVO);
 arduino.pinMode(7, Arduino.SERVO);
}

void captureEvent(Capture video) {
  // Läser kamerans bild
  video.read();
}

void draw() {
  video.loadPixels();
  image(video, 0, 0);

  // 500 som world record betyder att det är lätt att hitta en lägre
nummer
  float worldRecord = 500;

  //definerar integers
  int ClosestX = 0;
  int ClosestY = 0;
  int ArduinoX = 97;
  int ArduinoY = 110;
  int NollaX = 135;
  int NollaY = 100;
  int BromsX1 = 0;
  int BromsX2 = 0;
  int BromsX3 = 0;
  int BromsY1 = 0;
  int BromsY2 = 0;
  int BromsY3 = 0;
```

```
  int SnabbX = 0;
  int SnabbY = 0;

  // Börjar loopen att gå genom varje pixel i bilden.
  for (int x = 0; x < video.width; x ++ ) {
    for (int y = 0; y < video.height; y ++ ) {
      int loc = x + y*video.width;
      // Vilken är aktuella färgen
      color currentColor = video.pixels[loc];
      float r1 = red(currentColor);
      float g1 = green(currentColor);
      float b1 = blue(currentColor);
      float r2 = red(trackColor);
      float g2 = green(trackColor);
      float b2 = blue(trackColor);

 // Använder euklidiskt avstånd för att jämföra färgerna
      float d = dist(r1, g1, b1, r2, g2, b2);
// använder dist( ) för att jämföra track/currentColor.

      // sparar färgen om den ligger närmast trackColor
      if (d < worldRecord) {
        worldRecord = d;
        BromsX1 = ClosestX;
        BromsX2 = BromsX1;
        BromsX3 = BromsX2;
        ClosestX = x;
        SnabbX = ((ClosestX+BromsX3)/2) - ((BromsX1+BromsX2)/2);
        BromsY1 = ClosestY;
        BromsY2 = BromsY1;
        BromsY3 = BromsY2;
        ClosestY = y * (4/3);
        SnabbY = ((ClosestY+BromsY1)/2) - ((BromsY2+BromsY3)/2);

//om bollen rullar snabbt använd if satsen annars else

//styr X axeln
         if (SnabbX >=  2){

          if (ClosestX >= 320){//om bollen rullar sanbbt i X led
           ArduinoX = NollaX-(ClosestX/14)+8;}
            else{
           ArduinoX = NollaX-(ClosestX/14)-8;}
       }
        else{//Om bollen rullar långsamt i X led
              ArduinoX = NollaX-(ClosestX/14);
       }
 //Styr Y axeln
          if (SnabbY >=  2){  //om bollen rullanr snabbt i Y led
             if (ClosestY >= 320){
              ArduinoY = NollaY-(ClosestY/14)+8;}
            else{
           ArduinoY = NollaY-(ClosestY/14)-8;}
```

```
          }
 else{//Om bollen rullar långsamt i Y led
                ArduinoY = NollaY+(ClosestY/14);
          }

        }
    }
  }

  // använder color om skillnaden mindre än 10.
  if (worldRecord < 10) {
    // ritar en cirkel runt den följda pixeln
    fill(trackColor);
    strokeWeight(4.0);
    stroke(0);
    ellipse(ClosestX, ClosestY, 16, 16);


    arduino.servoWrite(6,ArduinoX);
    arduino.servoWrite(7,ArduinoY);

}
}

void mousePressed() {
  // sparar färgen där musen klickas
 int loc = mouseX + mouseY*video.width;
trackColor = video.pixels[loc];
}
```

Bilaga 2 Arduino IDE kod

```
// använder servo och firmata bibliotek
#include <Servo.h>
#include <Firmata.h>

Servo servo6;
Servo servo7;
//skriver värdet från PC till servo9 och servo10
void analogWriteCallback(byte pin, int value)
{
if(pin == 6)
servo9.write(value);
if(pin == 7)
servo10.write(value);
}
//gör det möjligt att ta emot data från PC
void setup()
{
Firmata.setFirmwareVersion(0, 2);
```

```
Firmata.attach(ANALOG_MESSAGE, analogWriteCallback);

Servo6.attach(6);
servo7.attach(7);

Firmata.begin(57600);
}
// gör en loop så länge PC:n skickar data
void loop()
{
while(Firmata.available())
Firmata.processInput();
}
```