Styrdosa till
Parans solpanel

Strand, Johan
johstr@student.chalmers.se
19870101–4899

Svedberg, Pär
svpar@student.chalmers.se
19821112–7652

Åkergren, Oskar
akergren@student.chalmers.se
19880508–7114

2015–03–23

Abstract

The purpose of this project is to find an alternative and more user friendly way of installing and controlling a sun panel manufactured by Parans Solar Lighting. As of today the panel is installed and controlled by connecting it to a computer, which has to be correctly configured and able to run a terminal where commands are sent to the panel. The aim of this project is to develop a hand held device which is easy to connect and has buttons that execute the commands needed to install and control the panel. The resulting device is a Raspberry Pi with an attached touchscreen, running a software developed in `Python`. The conclusion is that such a device is easily created with standard hardware, although this results in a large device compared to other touchscreen devices such as smartphones.

Sammandrag

Detta projekt ämnar hitta ett mer användarvänligt sätt att installera och styra en solpanel från Parans Solar Lighting. Idag utförs detta genom att ansluta panelen till en dator som måste vara korrekt konfigurerad och kunna köra ett terminalprogram för att skicka instruktioner till panelen. Målsättningen är att utveckla en handhållen enhet som är enkel att ansluta och har knappar som skickar instruktioner till panelen. Den enhet som tagits fram är en Raspberry Pi med pekskärm som kör mjukvara utvecklad i `Python`. Slutsatsen är att en sådan enhet med lätthet kan framställas med hjälp av existerande hårdvara. Detta resulterar dock i en större enhet jämfört med andra pekskärmsenheter såsom smarttelefoner.

Beteckningar

C	Imperativt programmeringsspråk
CP2102	Enhet från Silicon Labs som omvandlar kommunikation från USB till seriell enligt RS232
GUI	Graphical User Interface, grafiskt användargränssnitt
I/O	Input/Output
Java	Objektorienterat programmeringsspråk
PIC32	32-bitars mikrokontroller
Python	Högnivåspråk för programmering
SP3	Parans solpanel, tredje generationen
USB	Universal Serial Bus, standard för seriell kommunikation

Enkortsdator	I denna rapport menas enkortsdatorer av typen System on a chip, exempelvis enheter från Raspberry Pi och Beaglebone. En enkortsdator ska enligt rapportens definition klara av att driva operativsystem innehållande Linuxkärnan eller motsvarande.
Mikrokontroller	Här menas enchipdatorer avsedda att programmeras direkt till enhetens programminne, exempelvis enheter från Arduino. Dessa klarar ej av att driva operativsystem innehållande Linuxkärnan eller motsvarande.

Innehåll

Figurer

1 Introduktion

1.1 Bakgrund

Parans har utvecklat en produkt som via optiska fibrer levererar naturligt solljus. Som ett av få bolag i världen levererar de system globalt och deras för närvarande största installationer finns i Malaysia och Los Angeles.

Med hjälp av linser fokuseras solljus in i optiska fibrer och panelen styrs med hjälp av två stegmotorer. Styrningen sker på input dels från en algoritm som, baserat på position (longitud, latitud) och tid, ger en solposition i grader och dels från en solsensor med fotocell som ger data för en finstyrning av panelens positionering då solen är framme. Detta för att alltid maximera solljusets fokusering in i fibern.

Själva panelen drivs av en spänning om tolv (12) volt och dess systemdesign bygger på en PIC32; koden är skriven i `C`. Parans kommunicerar med enheten via USB–port och en terminalemulator.

1.2 Syfte

Vid installation och felsökning styrs panelen till rätt position via en terminalemulator i dagsläget, vilket är en tröskel för Parans kunder. Exempelvis har alla inte vana av att jobba i terminaler och det kan vara krångligt att konfigurera datorns USB–portar så att kommunikation kan ske med panelen.

Syftet med detta projekt är således att utveckla en produkt som underlättar vid installation och felsökningen av Parans solpaneler, genom att tillhandahålla ett gränssnitt som inte kräver någon djupare kunskap för att kunna använda.

1.3 Mål

Målet med projektet är att utveckla en produkt som uppfyller syftet i form av en styrdosa/box med tryckknappar, lysdioder och eventuellt en display som minskar problemen för kunderna. Denna box kan vara i form av ett befintligt kort som t.ex. Raspberry Pi, Arduino eller liknande men skulle också kunna vara en applikation för Android/iOS som kan installeras på kundens mobila enhet.

1.4 Frågeställning

Rapporten ämnar att besvara följande frågeställningar:

- Vad styr valet av plattform för styrdosan?
- Blir styrdosan enkel att använda?
- Påverkar valet av plattform huruvida styrdosan blir kompatibel med framtida versioner av solpanelen?
- Vilket programmeringsspråk lämpar sig för styrdosans mjukvara?

1.5 Avgränsning

Vi ser att detta projekt kommer kunna skapas med existerande hårdvara i form av mikrokontrollerkort, telefoner eller enkortsdatorer. Detta ger att vi kommer att begränsa projektet till dessa former och inte utveckla ett eget mönsterkort.

Dagens paneler kan i dagsläget kommunicera med externa enheter via en USB–port men saknar övriga kommunikationsmöjligheter. Detta gör att projektet begränsas till kommunikation via en ansluten USB–kabel och att trådlös kommunikation ej är möjlig.

2 Metod

Metodavsnittet delas upp i en teoretisk metod och en praktisk metod som beskriver det praktiska arbetsättet och de verktyg som används under processen.

2.1 Teoretisk metod

Denna rapport är skriven utifrån en variant av forskningsmetoden Design Science Research (DSR) beskriven i "A design science research methodology for information systems research" [1].

Metoden beskriver forskningsprocessen uppdelad i fem faser; *Problem Analysis & Motivation, Design & Development, Demonstration, Evaluation* and *Communication*. I den första fasen, *Problem Analysis & Motivation*, identifieras problemområdet tillsammans med Parans och projektets krav med möjliga lösningar sätts upp.

Över de följande tre faserna, *Design & Development, Demonstration* and *Evaluation*, sker en iteration där *Design & Development* handlar om att ta fram en prototyp som i varje iteration demonstreras för Parans och utvärderas för att se hur väl den uppfyller de krav som sattes upp i *Problem Analysis & Motivation*.
Resultat presenteras slutligen i fasen *Communication* som dels en muntlig presentation och dels en skriven rapport.

En alternativ metod till DSR är 'Action Research' (AR) vars mål är att iterativt lösa ett problem med hjälp av en grupp av definierade metoder [2]. AR är väldigt lik DSR i sitt utförande [3] och båda metoderna skapar om kunskap om specifika situationer och problem. Anledningen att valet föll på DSR var att metoden strävar mot att utveckla och skapa artefakter, vilket går väl ihop med projektet om att utveckla en handhållen fysisk enhet. Med artefakt menas i det här sammanhanget en prototyp som evalueras hur väl den löser ett uppsatta problem.

2.2 Praktisk metod

Avsnittet behandlar projektets praktiska arbetsmetoder och de verktyg som används under utvecklingsprocessen.

2.2.1 Agilt arbetssätt

I enlighet med den teoretiska metoden DSR utförs arbetsgången agilt, dvs. i iterationer och med ett nära samarbete med kunden Parans. Från systemutvecklingsmetoden *Scrum* plockas konceptet *Product backlog* vilket innebär en samlingsplats för alla önskemål om förändringar och funktioner i produkten. Dessa förändringar och funktioner härrör från ett antal *User stories* som är ett begrepp för att i vardagligt språk beskriva vad en användare av systemet vill uppnå.

För att lättare åskådliggöra innehållet i backlogen används verktyget *Waffle.io* vilket ger tillgång till ett onlinebaserat virtuellt *Scrum board*. Ett Scrum board är en anslagstavla uppdelad i kolumner där backlogens innehåll är listat som poster i en kolumn och projektets medlemmar flyttar posterna till kolumnerna "Redo", "Pågående" och "Klar" beroende på vilken uppgift de arbetar på för stunden. Detta gör det lättare för alla projektets medlemmar att se vilken uppgift varje projektmedlem arbetar på just nu och uppgifterna i backlogen kan sorteras efter prioritet.

2.2.2 Utvecklingsmiljö

För utvecklingen av mjukvaran och rapportskrivande används texteditorn Sublime Text som utvecklingsmiljö. Sublime erbjuder ett tydligt användargränssnitt, är plattformsoberoende och erbjuder autokomplettering av ord som existerar i någon av de filer som är öppna i editorn. Detta snabbar avsevärt på både utveckling av programkod och dokumentering av projektet.Den plattformsoberoende miljön underlättar samarbete och felsökning i ett projekt där utvecklarna opererar under både Windows, Linux och OS X.

2.2.3 Versionshantering

All projektrelaterad kod, vilket inkluderar mjukvarukod, rapport och övrig dokumentation, förvaras centralt genom den webbaserade versionshanteringsplattformen *GitHub*. GitHub bygger på versionshanteringssytemet *Git*, som är är ett strikt kommandobaserat system, där utvecklaren ges möjlighet att återskapa tidigare versioner av programfiler, spåra ändringar i dessa och framförallt ger flera utvecklare möjlighet att parallellt arbeta mot samma källkodsfil. Utöver versionshanteringen erbjuder GitHub funktioner för att hantera och fördela uppgifter inom ett projekt, möjlighet att spåra buggar och att visuellt visa vem som producerat ett aktuellt avsnitt kod.

Den webbaserade plattformen möjliggör även att externa parter som kunden Parans och projektets handledare har möjlighet att följa projekt och produktutveckling på distans genom den öppna tillgången till källkod och rapport samt via sociala funktioner så som flöden och wikis.

3 Genomförande

Detta avsnitt avser behandla projektets utförande enligt den i föregående avsnitt beskrivna metoden DSR.

3.1 Problemanalys

Parans vision var att utveckla en mobil, handhållen enhet som via seriell kommunikation kan kommunicera med och agera fjärrkontroll till solpanelen SP3. I ett inledande skede diskuterades tekniska lösningar och vilka funktioner som var önskvärda från bolagets sida. Dessa funktioner var indelade i ett grundutförande och två nivåer av extrafunktioner.

Grundutförandet innehöll funktioner för att kunna skicka styrkommandon i syfte att justera panelen vertikalt och horisontellt, omstart, läsa av tids- och geopositionsinställningar och att kunna försätta panelen i installationsläge.

Extrafunktionerna innebar att ansluta en GPS-modul till fjärrkontrollen. Den första nivån av extrafunktioner var att kunna verifiera panelens geoposition medan den andra nivån bestod av att i panelen kunna ställa in tids– och geopositionsuppgifter, bägge med hjälp av information givet av GPS-modulen. Projektets mål var att inom den givna tidsramen utveckla en fjärrkontroll enligt grundutförande och om tid återstod efter detta att påbörja implementation av extrafunktioner.

3.2 Design och utveckling

3.2.1 Val av plattform

För att kunna konstruera den typ av fjärrkontroll som möter projektets krav var först ett beslut tvunget att tas om vilken teknisk plattform som skulle användas. De alternativ som diskuterades och jämfördes var Androidbaserade enheter, Arduinosystem och enkortsdatorer, främst Raspberry Pi. Den jämförelse kring respektive plattforms egenskaper redovisas i avsnitt 3.2.2.

Egenskaper som beaktades var utvecklingskomplexitet, användarvänlighet, kostnad och kompatibilitet med SP3s enhet för seriell kommunikation, CP2102. USB-anslutningar kräver att en av enheterna agerar värd (eng. 'host') och för att kunna kommunicera seriellt till CP2102 behöver plattformen ha stöd för 'USB-host', då SP3 saknar stöd för detta.

Bland Androidenheter, tabell 1, sågs fördelar i att de har en färdigutvecklad produkt innehållande pekskärm, komplett datorsystem, integrerat batteri och att drivrutiner finns tillgängliga till CP2102. Nackdelar var att de är relativt dyra, information om vilka enheter som stöder USB-host är bristfällig och att för att använda drivrutinen till CP2102 krävs i de allra flesta fall att denna integreras manuellt i en egenbyggd Androiddistribution [4].

Arduino och Raspberry Pi, har till viss del gemensamma fördelar och nackdelar, vilket syns i tabell 2 och 3. Båda plattformarna har stöd för USB-host, tillgång till mycket information då de är populära bland entusiaster och att det finns många utbyggnadsmoduler. Vad som kan ses som negativt är att ingendera levereras med skärm eller tryckknappar och att enheten kan bli otymplig vid anslutning av flera tilläggsmoduler. En fördel Arduino har gentemot Raspberry Pi är att den förstnämnda har lägre energiförbrukning medan den sistnämnda å andra sidan har drivrutiner till CP2102 integrerade i Linuxkärnan. Till Arduino måste alltså en drivrutin först programmeras.

3.2.2 Jämförelsetabeller

Tabell 1: Androidenhet

	Fördelar		**Nackdelar**
+	Pekskärm medför stor valfrihet i utförande av användargränssnitt	−	Otydligt vilka enheter som stöder USB-host
+	Etablerat OS	−	Relativt dyr
+	Stor skärmyta	−	Mer prestanda än nödvändigt
+	Drivrutiner till CP2102 existerar	−	Kräver egenbyggd Androiddistribution

Tabell 2: Raspberry Pi

	Fördelar		**Nackdelar**
+	God tillgång till information	−	Mer prestanda än nödvändigt
+	God tillgång till utbyggnadsmoduler	−	Saknar skärm och knappar i grundutförande
+	Drivrutiner till CP2102 i Linuxkärnan	−	Kan bli otymplig vid användande av många tilläggsmoduler
+	Lågt pris		

Tabell 3: Arduinosystem

	Fördelar		**Nackdelar**
+	God tillgång till information	−	Saknar skärm och knappar i grundutförande
+	God tillgång till utbyggnadsmoduler	−	Otymplig vid användande av många tilläggsmoduler
+	Låg energiförbrukning	−	Saknar drivrutiner till CP2102

3.2.3 Mjukvaruutveckling

Fjärrkontrollens mjukvara har i enlighet med vår metod utvecklats i iterationer där den första prototypen hade som mål att upprätta en seriell anslutning mot solpanelen. När väl anslutning skapats var nästa steg att kunna sända instruktioner för att få solpanelen att vrida sig i önskad riktning. I detta stadie fungerade mjukvaran på samma sätt (men med reducerad funktionalitet) som när användaren är uppkopplad mot solpanelen via en terminalemulator. Instruktioner skrevs via tangentbord och sändes som strängar enligt den formatering som den mottagande mjukvaran på solpanelen kräver.

Nästa steg i processen var att utveckla ett grafiskt användargränssnitt. För att ebjuda användaren ett lättanvänt gränssnitt som fungerar väl ihop med en resistiv pekskärm så lades fokus på att implementera stora tydliga tryckknappar. Mjukvaran i solpanelen gör så att panelen börjar vrida sig i en riktning när den får en motsvarande instruktion och fortsätter i den riktningen tills användaren ger en instruktion om att stoppa. Ett första steg var då att utveckla knappar som skickade instruktioner för panelens olika riktningar och en knapp som stoppade all rörelse. När detta fungerade tillfredsställande så ändrades funktionaliteten till att panelen rör sig så länge användaren håller knappen nedtryckt men stannar när knappen släpps upp. Instruktionen för att stoppa panelen skickas således automatiskt när en knapp släpps upp och gör den dedikerade stoppknappen överflödig men det beslutades ändå att behålla den i händelse av att en extra stoppinstruktion behöver skickas.

Utöver funktionalitet för att manuellt styra panelen i önskad riktning har solpanelen instruktioner för att automatiskt lokalisera solen, starta om mjukvaran, försätta panelen i installationsläge och skriva ut information som tid, datum och geoposition. Då pekskärmens yta är begränsad beslutades att inte lägga alla motsvarande knappar för dessa funktioner i samma vy utan att separera den del som manuellt manövrerar solpanelen från övriga funktioner.

Efter att solpanelen kunde styras med hjälp av det grafiska gränssnittet var nästa steg att läsa in den information som panelen skickar som feedback till användaren. Denna information skickas som en samling strängar och innehållet består av den instruktion som användaren har skickat och åt vilket håll panelen rör sig åt. Informationen skrivs vid användning av terminalemulator ut rad för rad men eftersom vi med pekskärmen hade en begränsad yta att presentera informationen på så beslutades att implementera ett tvåradigt statusfält i botten på displayen. Detta statusfält är tillgängligt oavsett vilken vy användaren befinner sig. Här behövdes relevant information om panelens status filtreras ut så att användaren inte behöver ta del av de instruktioner som skickas fram och tillbaka mellan fjärrkontrollens mjukvara och mjukvaran i solpanelen.

4 Resultat

I det här avsnittet kommer resultatet att presenteras genom en redovisning av den produkt som genomförandet har lett fram till och vidare avser avsnittet att knyta an till frågeställningen och besvara de frågor som ligger till grund för arbetet.

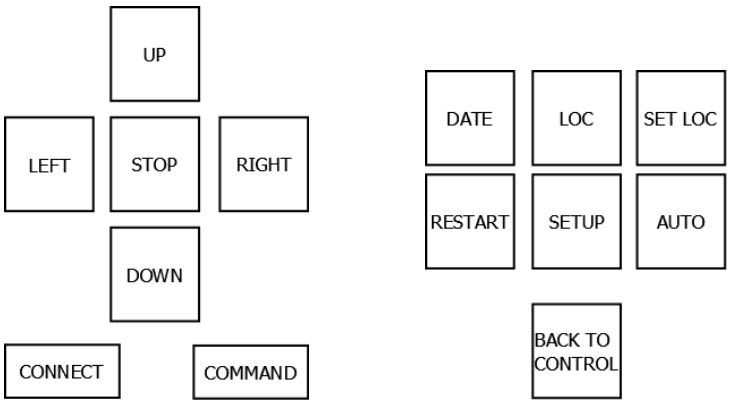
4.1 Hårdvara

Den fysiska produkt som genomförandet resulterade i, är en handhållen produkt som kopplar in sig till solpanelen via en USB–sladd. Grunden i enheten är en enkortsdator av märket 'Raspberry Pi' (se figur 2, bilaga A) och användargränssnittet består av en resistiv pekskärm framtagen till just denna plattform [5] (se figur 3). Produkten placeras i ett inköpt chassi framtaget för att husera just denna konfiguration. Enheten strömförsörjs genom ett batteripaket, även det inhandlat från återförsäljare, som är framtaget för att agera laddare för enheter som laddas via USB. Denna typ av laddningsenhet visade sig vara lämplig även för vår produkt.

4.2 Mjukvara

Den mjukvara som utvecklats till enheten är skrivet i programmeringsspråket `Python` och består av en anpassad stränghantering och ett grafiskt gränssnitt uppbyggt av det medföljande paketet 'TkInter'. För en översikt av mjukvarans klasser, se figur 4. Stränghanteringen är anpassad till det sätt som SP3 hanterar textsträngar, så att texten visas upp på ett korrekt sätt när den mottagits över den seriella kommunikationen. Gränssnittet är uppdelat i två primära vyer, se figurerna 1a och 1b, där de olika knapparna genererar de instruktioner som solpanelens styrkort lyssnar efter.

Den första vyn som öppnas när applikationen startar är den i 1a och innehåller de grundläggande styrfunktionerna för justering av solpanelens rotering i X– och Y–led. Knappen 'command' leder till den andra vyn som visas i figur 1b. Den andra vyn hanterar de olika instruktioner som vanligen används för felsökning av SP3 så som att hämta satt datum, satt plats och omstart av enheten.



(a) Styrning, uppkoppling & genväg till 1b

(b) Instruktionsgränssnittet 'command'

Figur 1: Skiss av det grafiska gränssnittet

4.3 Frågeställningen

Vad styr valet av plattform för styrdosan? Valet av plattform grundar sig i den jämförelse mellan de olika plattformarna som redovisas i avsnitt 3.2.1 där beslutet fattades främst efter motiveringarna *Tillgänglighet* och *Utvecklingskomplexitet*, då projektet utförts inom en snäv tidsram och en vilja att kunna leverera en fungerande produkt till uppdragsgivaren.

Enkel för kunderna att använda? Projektet resulterade i en produkt som är att anse som användarvänlig, i synnerhet när produkten sätts i förhållande till den nuvarande tillvägagångssättet. Produktens användargränssnitt påminner om andra gränssnitt avsedda för styrning av andra produkter. Vår produkt vänder sig till montörer av solpanelen, så viss kunskap om panelen krävs för att kunna nyttja enheten till fullo.

Framtidskompatibel? Den produkt som producerats är framtidskompatibel, då den baseras på ett huvudkort som har flertalet kommunikationsportar och med en Linuxdistribution i grunden är mjukvaran lätt att justera efter behov. Kompatibiliteten bryts då en framtida generation av solpaneler använder sig av en kommunikationsstandard som inte finns integrerad på Raspberry Pi, men i dagsläget finns inga sådana planer.

Programmeringsspråk Valet av programmeringsspråk grundar sig i det metodval projektet har arbetat efter, där ett fokus ligger på att producera prototyper och där `Python` är, sett ur programmeringsperspektiv, ett effektivt programmerings språk där en fungerande mjukvara snabbt kan tas fram. `Python` bidrar även med en lättöverskådlig programmeringssyntax som underlättar projektet framtida utveckling och dess portabilitet gör att mjukvaran enkelt kan flyttas till en annan plattform om ett sådan behov uppstår.

5 Diskussion

Syftet för projektet var att ”utveckla en produkt som underlättar installation och felsökning, genom att tillhandahålla gränssnitt som inte kräver någon djupare kunskap för att kunna använda” vilket projektet även har resulterat i. För de mest grundläggande funktionerna, att kunna styra panelen i X- och Y-led, använder vi ett utseende på gränssnittet som påminner om ett styrkors vilket de flesta personer har någon form av erfarenhet av, då sådana återfinns på diverse olika typer av fjärrkontroller. De övriga instruktionerna som finns tillgängliga i gränssnittet är inte helt självförklarande för en icke insatt person, men de installatörer som ska nyttja vår produkt bör ha en uppfattning om vad de instruktionerna gör, särskilt med stöd från Parans. Framförallt ser vi att denna produkt är betydligt enklare att använda än att koppla upp en dator med en terminal emulator.

Målet med projektet var att framställa en produkt som skulle kunna nyttja det gränssnitt som syftet efterfrågade och skulle fungera som en styrdosa eller trådbunden fjärrkontroll. Vi anser att detta projekt har uppnått det målet och produktens olika egenskaper kommer att diskuteras under efterföljande rubriker.

5.1 Hårdvara

Som nämnt på sidan ii så utgår vi ifrån begreppet enkortsdator för ett kretskort som är kapabel till att driva en Linuxkärna, till skillnad från en mikrokontroller där en svagare krets avses.

Vår lösning är baserat på en enkortsdator och är fullt fungerande enligt de krav som uppdragsgivaren har fastställt och är relativt enkel att reproducera, i förhållande till att utveckla en likartad konstruktion med en mikrokontroller. Det som gör vår lösning enklare är framförallt att en enkortsdator har de drivrutiner som krävs för att upprätta den seriella kommunikationen, så till vida att den har en Linuxkärna senare än version 3.0 [6].

Nackdelar som vi ser med att använda en enkortsdator är bland andra att dessa generellt har ett större energibehov än en mikrokontroller [7, 8]. Antalet I/O portar är oftast färre på en enkortsdator och den fysiska storleken är större jämfört med de mikrokontrollerkort som hade varit lämpliga för projektet.

Gällande frågeställningen om vår produkt är enkel att använda för kunderna, så ger vår produkt ett enkelt gränssnitt att använda, men en annan produkt med pekskärm kan komma att upplevas som lika enkel. Vår produkt är något klumpig, vilket vi även påtalar i avsnitt 3.2.1, något som kan påverka användarvänligheten. En Androidenhet kan vara enklare att greppa om och visa upp samma gränssnitt, så ur en användares synsätt kan vår produkt inte vara den enklaste att nyttja, men ur en utvecklares perspektiv är det svårare att forma Android att göra det vi vill, så projektet skulle kunna ha resulterat utan någon produkt överhuvudtaget.

5.2 Mjukvara

Den mjukvara som har utvecklats, har skrivits i programmeringsspråket `Python`. Språkvalet beror delvis på att personal inom företaget har erfarenhet inom språket vilket underlättar för framtida utveckling och underhåll av projektets produkt och dels valdes språket för dess enkla utveckling av grafiska gränssnitt och bra stöd i den seriella kommunikationen som krävdes i projektet.

Andra språk som hade varit möjliga är till exempel `C` eller `Java` då projektgruppen har erfarenhet av de båda språken. `C` valdes bort då utveckling av grafiska gränssnitt i detta språk kräver externa bibliotek och minskar därför portabiliteten och ökar komplexiteten. `Java` är en lämplig kandidat för projektet, men valdes bort då den grafiska utvecklingen i `Python` är enklare och applikationen som vi utvecklade är såpass simpel att `Java` skulle medföra stor andel så kallad 'overhead' i programmeringskoden, något som visas när en jämförelse görs mellan de olika språken.[9] Nackdelen med `Python` jämfört med `Java` är att språket inte är lika effektivt i sina beräkningar, men då applikationen vi skrivit inte utför några tyngre beräkningar så berörs inte användarupplevelsen av detta. Kommer applikationen att vidareutvecklas till något mer än vad projektet skapat, är det fullt rimligt att översätta logiken till `Java`, något som det finns gott om stöd för.[10]

5.3 Framtida bruk

Att projektet genomfördes grundar sig i SP3s bristande stöd för kommunikationsstandarder och att dagens kommunikationsgränssnitt inte är användarvänligt, vilket leder till stora underhållskostnader för företaget då det krävs tid och resurser att stötta underhållspersonal. Detta projekt svarar upp på de förväntningar som bolaget hade på oss, men vi ser att projektets produkt kan komma att bli överflödig i nyare revisioner av panelen, där styrkortet kan ha tillgång till fler kommunikationsstandarder och kan komma att styras på distans.

Referenser

[1] K. Peffers, T. Tuunanen, M. A. Rothenberger och S. Chatterjee. "A Design Science Research Methodology for Information Systems Research". I: *Management Information System* 24 (3 2007), s. 45–78. URL: http://wise.vub.ac.be/thesis_info/Design_Science_Research_Methodology_2008.pdf (hämtad 2015-02-04).

[2] K. Hinkelmann och H. F. Witschel. "How to choose a research methodology?" 2013. URL: http://knut.hinkelmann.ch/lectures/project2013/p1_5_how-to-choose-a-research-methodology.pdf (hämtad 2015-02-27).

[3] R. Baskerville, J. Pries-Heje och J. Venable. "Soft Design Science Methodology". I: *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology* (2009). URL: http://dl.acm.org/citation.cfm?id=1555631 (hämtad 2015-02-04).

[4] Silicon Laboratories Inc. *Integrating the CP210X Virtual COM Port Driver into the Android Platform*. 2014. URL: https://www.silabs.com/Support%20Documents/TechnicalDocs/an809.pdf (hämtad 2015-02-27).

[5] L. ADA. *Adafruit PiTFT – 2.8Touchscreen Display for Raspberry Pi*. 2015. URL: https://learn.adafruit.com/adafruit-pitft-28-inch-resistive-touchscreen-display-raspberry-pi/overview (hämtad 2015-02-27).

[6] Silicon Laboratories Inc. *CP210x USB to UART Bridge VCP Drivers*. 2015. URL: https://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx (hämtad 2015-02-04).

[7] Igor. *Arduino Power Consumption Normal & Sleep*. 2013. URL: http://gadgetmakersblog.com/arduino-power-consumption/ (hämtad 2015-02-04).

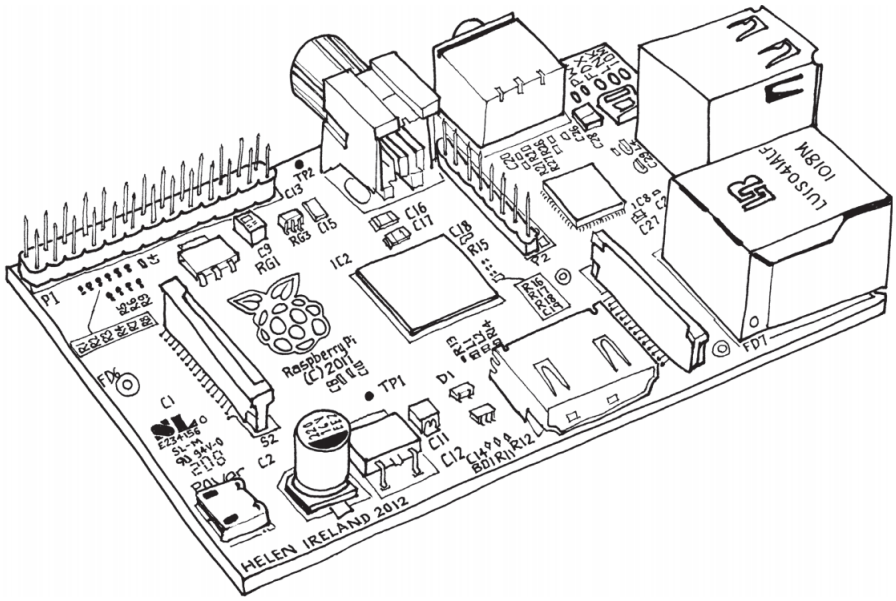
[8] Raspberry Pi Foundation. *Power Supply*. 2015. URL: http://www.raspberrypi.org/documentation/hardware/raspberrypi/power/README.md (hämtad 2015-02-04).

[9] S. Ferg. *Python & Java: A Side-by-Side Comparison*. 2011. URL: https://pythonconquerstheuniverse.wordpress.com/2009/10/03/python-java-a-side-by-side-comparison/ (hämtad 2015-02-27).

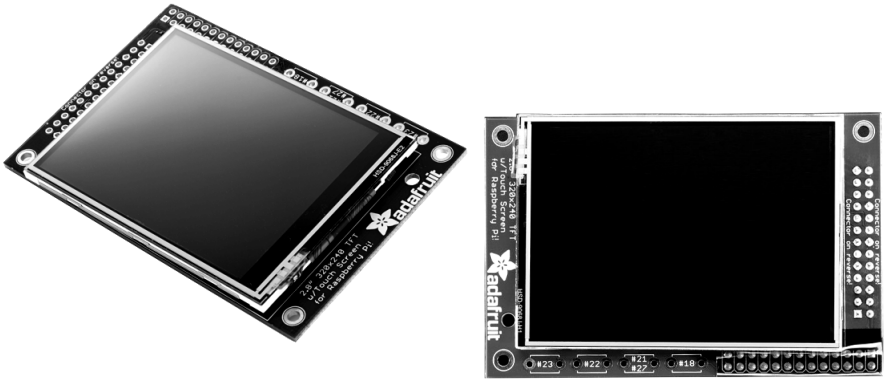
[10] Jython. *General Information*. 2014. URL: https://wiki.python.org/jython/JythonFaq/GeneralInfo (hämtad 2015-02-25).

Appendix

A Komponenter

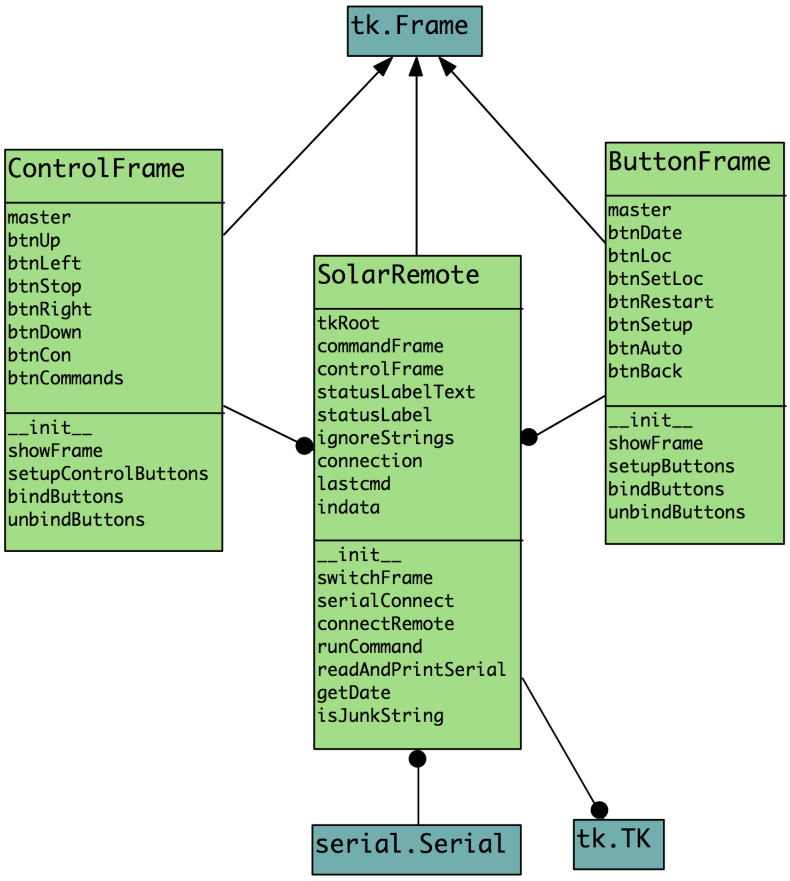


Figur 2: Raspberry Pi
http://pi.cs.man.ac.uk/download/Raspberry_Pi_Education_Manual.pdf



Figur 3: Pekskärm
http://www.linuxuser.co.uk/features/10-raspberry-pi-upgrades-part-2
http://www.adafruit.com/product/1601

B UML



Figur 4: UML–diagram av mjukvaran