

LUNDS TEKNISKA HÖGSKOLA

PROJEKTRAPPORT ELEKTRONIKPROJEKT OCH
HÅLLBAR UTVECKLING FÖR E3

ESSF05

BigBroRoto

Författare:

Alexander NAJAFI
Henrik FELDING
Hugo HJERTÉN
Linus HELLMAN
Tobias MÄHL

Handledare:

Mikael SWARTLING

25 maj 2015

Sammanfattning

I projektet BigBroRoto konstruerades ett system vars funktion var att beräkna en ljudkällas position utifrån detta systemet och sedan vrida en kamera mot dena källa. Detta system konstruerades runt en DSP (Digital Signal Processor) där algoritmen som beräknar ljudets ingångsvinkel utifrån tidsförskjutningarna i de ingående ljudsignalerna implementerades. Dessa fyra ljudsignaler uppkom i fyra mikrofoner som satt placerade i en kvadratisk formation. När vinkeln, som kontinuerligt uppdaterades, var uträknad skickas denna till en Raspberry Pi som behandlade vinkeln. Behandlingen gjordes så att kameran inte skulle röra sig alltför ryckigt och för att den inte skulle göra alltför små justeringar. När behandlingen var klar skickades vinkeln via ett HTTP anrop till kameran som styrde in sig i rätt riktning. På Raspberry Pi:n implementerades en webserver så att systemet fick ett stilrent front end där systemet kunde styras, där information om nuvarande vinkel kunde läsas och där man även kunde se videoströmmen från kameran. Projektet slutfördes inom tidsramen och fungerade mer eller mindre som önskat.

Summary

In the project BigBroRoto a system was created whose function was to calculate the position of a sound source and then rotate a camera towards this source. This system was constructed around a DSP (Digital Signal Processor) whose function was to calculate the angle of the incoming sound based on the time offsets in the four input audio signals. These four audio signals were created from four microphones that were positioned in a quadratic array. When this angle, which was updated continuously, was calculated it was sent to a Raspberry Pi that processed the angle. The angle was processed to prevent the camera from moving too erratically and to prevent it from making too small adjustments. When the processing of the angle was finished it was sent via HTTP to the camera that turned it in the right direction. On the Raspberry Pi a web server was implemented so that the system received a stylish front end where the system could be controlled, one could get information about the current angle and where one could see the video stream from the camera. The project was completed on time and worked more or less like initially planned.

Förord

En maskin som kan lokalisera vartifrån ett ljud kommer ifrån, hur häftigt är inte det? Vår ursprungliga inspiration till BigBroRoto kom från den s.k Popinatorn; en maskin som reagerar på ordet 'pop' och skjuter ett popcorn dit ljudet kom ifrån. I detta projekt använder vi istället ljudlokaliseringmekanismen för att styra en kamera så att den pekar mot ljudkällan. Detta skulle tex vara mycket praktiskt vid en videokonferens!

Vi vill lyfta ett stort tack till Mikael Swartling; forskare, TeknD och vår handlare i detta projekt. Med Mikaelns spetskunskaper från hans doktorsavhandling¹ fick vi ovärderliga tips och vägledning. Vi fick även testa BigBroRoto i det helt ekoisolerade rummet, häftigt! Tack Mikael!

¹Direction of Arrival Estimation and Localization of Multiple Speech Sources in Enclosed Environments, Mikael Swartling, Blekinge Tekniska Högskola, 2012

Innehåll

1 Inledning	4
2 Teori	4
2.1 System med två mikrofoner	4
2.2 System med fyra mikrofoner	5
2.3 Kommunikation	6
2.3.1 SPI	6
2.3.2 DSP	7
2.3.3 Axis HTTP API	7
2.4 Raspberry Pi	7
2.4.1 Front end	7
2.4.2 Daemon	8
3 Genomförande	8
3.1 Tillvägagångssätt	8
3.2 Hårdvara	8
3.3 Mjukvara	9
3.3.1 Signalbehandlingsalgoritm	9
3.3.2 Kommunikation	9
4 Resultat och diskussion	10
4.1 Filtrering i Raspberry Pi:n	10
4.2 Användarupplevelse	11
4.3 Flera ljudkällor	11
5 Återblick	12
6 Slutsatser och fortsatt arbete	12
6.1 Viktning	13
6.2 Sökning i höjdled	13
A Appendix	15
A.1 Mötesprotokoll	15
A.2 GIT-repo	17
A.3 Huvudkod i DSP:n	17

1 Inledning

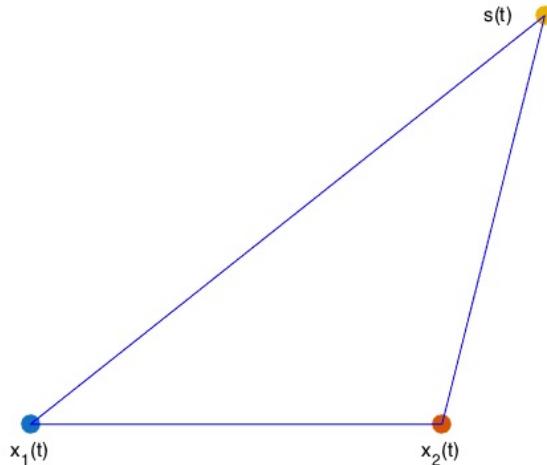
BigBroRoto är ett system som kan styras och filmar i riktning mot ljudkällan. Detta görs genom att ha 4 mikrofoner i en kvadratisk formation som alla mäter ljud i realtid. Genom en avancerad algoritm i en DSP kan en infallsvinkel räknas ut, som sedan skickas vidare och behandlas av en Raspberry Pi. Med den informationen styr Raspberry Pi:n sedan kameran så att den pekar mot ljudkällan.

Denna rapport ämnar förklara algoritmen som ligger bakom beräkningen av infallsvinkeln samt de kommunikationsmetoder som används mellan de olika systemen i projektet. Rapporten förklrar även tillvägagångssättet, varför de olika systemen valdes och hur de olika delmomenten sattes ihop. Slutligen diskuteras den färdiga 'produkten' och huruvida målet att framställa en kamera som vrider sig till den som talar uppnådes.

2 Teori

2.1 System med två mikrofoner

Om två mikrofoner är placerade med att känt inbördes avstånd så kommer ljud som utbreder sig i rummet att anlända till mikrofonerna med en tidskillnad som beror på ljudkällan och mikrofonernas positioner.



Figur 1: Två mikrofoner, $x_1(t)$ & $x_2(t)$, och en ljudkälla, $s(t)$.

Det gäller att

$$x_1(t) = s(t + T) \quad (1)$$

där T är propageringstiden från källan till mikrofon 1. Det gäller också att

$$x_2(t) = s(t + T + \tau) \quad (2)$$

där T är propageringstiden till mikrofon 1 och τ är tidskillnaden från det att ljudvågen når mikrofon 1 tills det att den når mikrofon 2

Genom Fouriertransformation fås följande:

$$X_1(\omega) = \int_{-\infty}^{\infty} x_1(t) \cdot e^{-j\omega t} dt = \int_{-\infty}^{\infty} s(t+T) \cdot e^{-j\omega t} dt = S(\omega) \cdot e^{j\omega T} \quad (3)$$

på samma sätt fås

$$X_2(\omega) = \int_{-\infty}^{\infty} x_2(t) \cdot e^{j\omega t} dt = \int_{-\infty}^{\infty} s(t+T+\tau) \cdot e^{-j\omega t} dt = S(\omega) \cdot e^{j\omega(T+\tau)} \quad (4)$$

Det gäller att $x_2(t) = x_1(t + \tau)$. Genom att beräkna korrelationen mellan $x_1(t)$ och $x_2(t)$ fås följande uttryck.

$$r_{x_1, x_2}(t) = \mathcal{F}^{-1}\{R_{X_1, X_2}\} = \mathcal{F}^{-1}\{X_1(\omega) \cdot \bar{X}_2(\omega)\} \quad (5)$$

Detta kan förenklas till

$$r_{x_1, x_2}(t) = \mathcal{F}^{-1}\left\{S(\omega) \cdot e^{j\omega T} \cdot S(\bar{\omega}) \cdot e^{-j\omega(T+\tau)}\right\} = \mathcal{F}^{-1}\{|S(\omega)|^2 \cdot e^{-j\omega\tau}\} \quad (6)$$

$$r_{x_1, x_2}(t) = \mathcal{F}^{-1}\{|S(\omega)|^2 \cdot e^{-j\omega\tau}\} = \int_{-\infty}^{\infty} |S(\omega)|^2 \cdot e^{-j\omega\tau} \cdot e^{j\omega t} d\omega \quad (7)$$

en skattning av den tid τ som ger störst korrelation mellan signalerna ges då av.

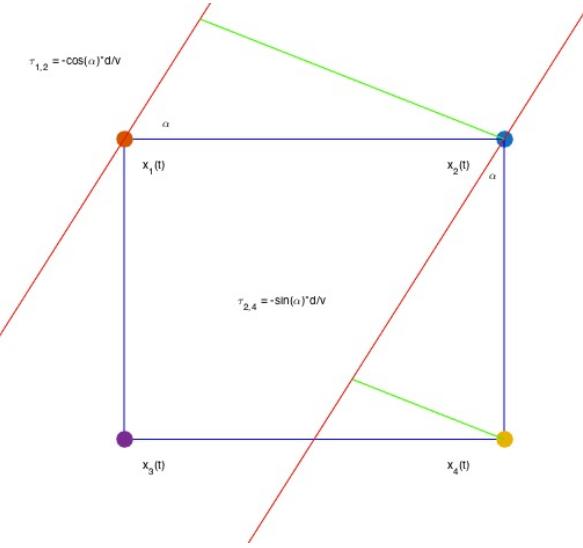
$$\hat{\tau} = \arg \max \tau' \int_{-\infty}^{\infty} |S(\omega)|^2 \cdot e^{-j\omega\tau} \cdot e^{j\omega\tau'} d\omega \quad (8)$$

Ekvationen ovan går ej att lösa analytiskt utan man måste maximera uttrycket genom att använda numeriska metoder.

2.2 System med fyra mikrofoner

Det går att utöka systemet beskrivet i stycket ovan genom att använda fler mikrofoner. För att relatera tidskillnaden mellan de olika mikrofonerna till infallsvinkelns α införs följande modell.

Om approximationen att en infallande ljudvågen är plan görs så kan följande uttryck för den tid det tar från att ljudvågen har nått den första mikrofonen till det att den når den andra mikrofonen ställas upp.



Figur 2: Fyra mikrofoner $[x_1(t), x_2(t), x_3(t), x_4(t)]$ och inkommande ljud från en ljudkälla.

Ur denna modell kan följande ekvationer härledas.

$$\begin{cases} \tau_{1,2}(\alpha) = -\cos(\alpha) \cdot \frac{d}{v} \\ \tau_{2,4}(\alpha) = -\sin(\alpha) \cdot \frac{d}{v} \\ \tau_{4,3}(\alpha) = \cos(\alpha) \cdot \frac{d}{v} \\ \tau_{3,1}(\alpha) = \sin(\alpha) \cdot \frac{d}{v} \end{cases} \quad (9)$$

Ekvation 8 kan då generaliseras till

$$\hat{\alpha} = \arg \max \alpha \int_{-\infty}^{\infty} \sum_{i=1}^4 G_i \cdot e^{j\omega\tau_i(\alpha)} \quad (10)$$

där

$$\begin{cases} G_1 = X_1(\omega) \cdot \bar{X}_2(\omega), \tau_1(\alpha) = \tau_{1,2}(\alpha) \\ G_2 = X_2(\omega) \cdot \bar{X}_4(\omega), \tau_2(\alpha) = \tau_{2,4}(\alpha) \\ G_3 = X_4(\omega) \cdot \bar{X}_3(\omega), \tau_3(\alpha) = \tau_{4,3}(\alpha) \\ G_4 = X_3(\omega) \cdot \bar{X}_1(\omega), \tau_4(\alpha) = \tau_{3,1}(\alpha) \end{cases} \quad (11)$$

2.3 Kommunikation

2.3.1 SPI

Kommunikationen mellan DSP:n och enkortsdatorn, Raspberry Pi:n, sker över en buss kallad SPI (Serial Peripheral Interface). I implementationen av BigBro-Roto används fyra trådar för kommunikationen och det fanns bara två enheter som skulle kommunicera med varandra. Den ena enheten är en master-enhet som styr och bestämmer allt om kommunikationen (Raspberry Pi) medan den

andra enheten är en slav-enhet (DSP). Två av trådarna i SPI används för dataöverföringen, trådarna MOSI (Master out, slave in) och MISO (Master in, slave out). En tråd SS (Slave select) skulle egentligen inte behövas i en implementering av två enheter, då den används för att bestämma vilken enhet som ska kommunicera med master-enheten. Den fjärde tråden används för synkronisering, och är alltid en klocksignal som bestämmer när enheterna ska skicka sina bitar.

I implementationen av BigBroRoto behöver endast DSP:n skicka data till RaspberryPi:n i form av uträknade vinklar. Det är dock som sagt Raspberry Pi:n som avgör när data ska skickas. När Raspberry Pi:n vill ta emot en ny vinkel från DSP:n så börjar den med att sätta SS pinnen till en digital nolla (SS pinnen är aktiv låg). Efter detta startar Raspberry Pi:n en klocksignal på CLK-pinnen. För varje klockpuls vet då DSP:n att den ska sätta en ny bit på MISO-pinnen som kan läsas av Raspberry Pi:n. När åtta bitar klockats igenom höjer Raspberry Pi:n SS pinnen tillbaka till en digital etta och ett paket om åtta bitar har förts över.

2.3.2 DSP

DSP-enheten är konfigurerad som slav-enhet och blir alltså tillfrågad när den ska skicka data. I processorn är SPI-protokollet hårdvaruimplementerat för en effektiv överföring. När algoritmen har räknat ut en ny vinkel skrivas denna vinkel som åtta bitar till ett register (TXSPI) i processorn. När ny data skrivas till detta register skrivas automatiskt datan över i ett skiftregister. DSP:n konfigurerades så att datan ligger kvar i detta skiftregister ända tills det skrivas över med ny data från TXSPI registret.

När SS-pinnen blir aktiv och en klocksignal kommer in på CLK-pinnen börjar bitarna från skiftregistret att klockas ut på MISO-pinnen. Skulle ny data skrivas till TXSPI under denna tiden väntar processorn med att skriva över bitarna till skiftregistret tills överföringen är klar.

2.3.3 Axis HTTP API

I de flesta moderna Axis-kameror som är IP-baserade finns en liten enkel inbyggd webbserver. Denna används för att kunna kommunicera över HTTP med kameran och möjliggör styrning av nästan alla kamerans funktioner och konfigurationer. Raspberry Pi:n och Axis-kameran kan kommunicera med varandra över en router som routar och styr trafiken. I Axis kamerans webbserver finns en implementation av ett HTTP API. Detta används av Raspberry Pi:n som gör HTTP GET anrop till kameran för att styra den.

2.4 Raspberry Pi

2.4.1 Front end

Raspberry Pi:n drivs av Debian-version som är speciellt skapat för Raspberry Pi:s ARM-arkitektur. På Raspberry Pi:n körs en webbserver av typen Apache2 med tillägget PHP-FPM som möjliggör skriptning av PHP-sidor. Detta användes för att skapa en hemsida där användaren av BigBroRoto kunde styra systemet

och kameran. I figur 4.2 synes gränssnittet i en webbläsare. I gränssnittet kan användaren se bilden från kameran, styra zoom och höjd, se en liveuppdatering av loggar och se den senaste vinkeln som DSP:n räknat ut.

Kommunikationen mellan användarens webbläsare och DSP:n som räknar ut vinklar är lite komplex. DSP:n skickar som beskrivet tidigare vinklar över SPI till Raspberry Pi:n. På Raspberry Pi:n körs en daemon som öppnar upp en socket-server för anslutning av klienter. Daemonen matar ut loggar och nya vinklar till alla klienter som ansluter till socket-servern. För att klienterna som ansluter till webbservern ska kunna ta del av datan så öppnar webbservern en socket-anslutning till daemonen när en användare försöker ladda gränssnittet. Webbservern matar sedan ut en ström av data över HTTP:s SSE (Server sent events) gränssnitt. Detta plockas upp av webbläsaren som visar logg- och vinkeldata i realtid för användaren.

2.4.2 Daemon

I bakgrunden på Raspberry Pi:n körs en daemon som sköter hela BigBroRoto. Daemonen har flera trådar där en tråd sköter kommunikationen över SPI med DSP:n och en tråd som sköter socket-hanteringen för utmatning av vinklar. Det är denna daemonen som startas och stoppas av start- och stoppknapparna i webbgränssnittet. Tråden som sköter kommunikationen med DSP:n frågar DSP:n efter en ny vinkel tre gånger varje sekund och sköter en filtrering av vinklarna. Det är även denna tråd som kommunicerar med kameran över HTTP.

3 Genomförande

3.1 Tillvägagångssätt

Under projektets gång delades gruppen upp i två huvudområden; en mjuk- och en hårdvarugrupp. Den ena gruppens ansvar var att skriva en fungerande algoritm i MatLab för att sedan implementera den i C-kod för DSP:n. Den andra gruppen såg till att kommunikationen mellan de olika systemen fungerade, hittade de korrekta pinnarna på DSP:n, lärde sig DSP:ns simulerings- och emuleringsmiljö², konstruerade hemsidan och kodade Raspberry Pi:n, och byggde den kvadratiska mikrofonstrukturen. Grupperna kunde jobba parallellt tills delmomenten var färdiga. Tiden då algoritmen skulle implementeras i C-kod och simuleras blev den naturliga tidpunkten då de två delgrupperna arbeten sammansvetsades.

Under projektets gång hölls möten med handledaren Mikael ungefär en gång i veckan.

3.2 Hårdvara

Första steget i hårdvaruimplementeringen av BigBroRoto var att driva kameran. Kamerans spänningsadapter saknades så två sladdar löddades direkt på kamerans moderkorts matningspunkter. Ett hål borrades sedan i kamerans chassi

²Analog Devices VisualDSP++ Environment

så att kablarna snyggt kunde dras ut och spänningssättas med en 12V DC-transformator. Kameran och Raspberry Pi:n kopplades samman med vanliga ethernetkablar via en router.

Ljudet in till DSPn valdes att samlas upp utav fyra kondensatormikrofoner. Sma mickar valdes på grund av precisionsskäl. En annan faktor i valet av mikrofoner var att de skulle kunna drivas av DSPns mikrofonformatning. Mikrofonerna löddes fast på ett labbkort i en kvadratisk formation med sidorna 39mm. Mikrofonerna kopplades sedan in i på DSPns ljudingångar.

Det sista steget i hårdvaruimplementationen av BigBroRoto var att montera allt på ett träblock så att mikrofonerna satt upphöjt över möjliga hinder för ingångsljudet. Sedan fästes resten av komponenterna fast i samma träblock så att allt fanns samlat och kunde presenteras snyggt.

3.3 Mjukvara

3.3.1 Signalbehandlingsalgoritm

Efter att ha tagit fram en teoretisk modell för hur stor tidskillnaden mellan de olika mikrofonerna skulle vara så gjordes en simulering av den teoretiska modellen i MatLab. För att verifiera att den teoretiska modellen fungerade gjordes tester dels med vitt brus, dels med en inspelad ljudfil. För att undersöka kunna undersöka signalbehandlingsalgoritmens funktionalitet skrevs en funktion som gjorde om en enkanals .wav-fil till en matris med fyra koloner där kolonerna hade blivit förskjutna i förhållande till varandra enligt den teoretiska modellen. Efter att ha bekräftat att den teoretiska modellen fungerade så implementerades signalbehandlingsalgoritmen i C med hjälp av utvecklingsmiljön Analog Devices Visual DSP++, den utvecklingsmiljö som följe med DSPn. I Visual DSP++ fanns ett bibliotek med många användbara funktioner för signalbehandling, bland annat Snabb Fouriertransform och matematikoperationer för komplexa tal.

3.3.2 Kommunikation

Innan arbetet med att få alla enheter att kommunicera kunde påbörjas undersöktes möjligheterna om för att implementera SPI på DSP:n. Tester om huruvida rätt pinnar fanns utdragna från processorn gjordes och olika konfigurationer testades. Först när SPI på DSP:n fungerade påbörjades arbetet med att implementera SPI på Raspberry Pi:n. DSP:n konfigurerades som slav-enhet och kopplades ihop med Raspberry Pi:n som konfigurerades som master-enhet. Efter mycket felsökning med oscilloskop kunde rätt pinnar kopplas ihop och rätt konfigurationer i båda enheterna göras. Enheterna kopplades ihop med SPI och med en gemensamt jordning. Enheterna testades med olika klockhastigheter och matchades ihop med en klockhastighet som båda enheterna klarade av att hantera. Storleken på paketen som skickades över SPI konfigurerades till åtta bitar per paket då detta var maxstorleken som Raspberry Pi:n kunde hantera.

När SPI fungerade påbörjades arbetet med kommunikationen med kameran. Med hjälp av bra dokumentation om VAPIX (API:t som möjliggör kommunika-

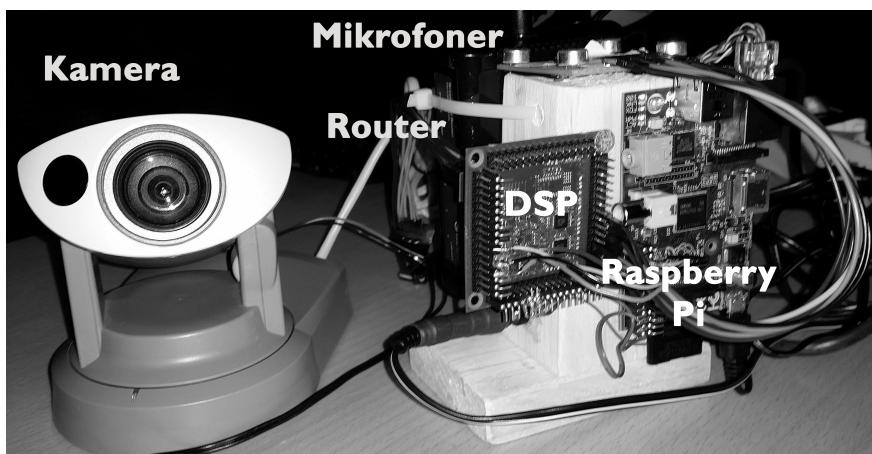
kation med Axis-kameran) kunde en klass skrivas i Python för att styra den.

Alla enheter kopplades ihop i ett nätverk med statiska IP-adresser med hjälp av en router. På så sätt kunde alla enheter nås på ett enkelt sätt från en dator som anslöts till nätverket.

4 Resultat och diskussion

Inför redovisningen den 19 maj kunde alla gruppmedlemmar stå i en halvcirkel och en i taget prata mot BigBroRoto så att kameran vred sig mot den talande med stor framgång. Det uppstod en fördräjning mellan en uppdaterad position av ljudkällan och kamerans rörelse på ungefär en sekund. När kameran hade hittat rätt så stod den oftast helt stilla i den positionen tills det uppstod en större förnyning i positionen av ljudkällan. Dock kunde det hända att kameran bara nästan pekade rätt och då stannade i den positionen och först uppdaterade sig vid ytterligare en större förändring av positionen.

Kamerans livefeed kunde vid redovisningen ses via BigBroRotos FrontEnd, den användarvänliga hemsidan där man även kunde styra höjd och in-zoomning av kameran. På hemsidan visades även sifervärdet av infallsvinkeln samt en pil som pekade i vinkelns riktning. Via hemsidan gick det att stänga av och på daemonen i Raspberry Pi:n.



Figur 3: BigBroRoto; den färdiga slutprodukten.

4.1 Filtrering i Raspberry Pi:n

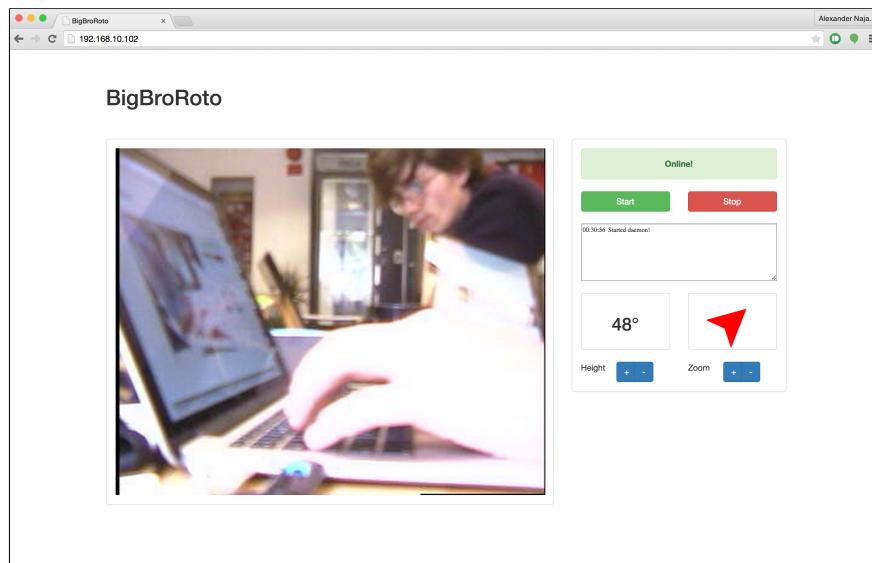
Ursprungligen så skickade Raspberry Pi:n enkelt vidare den uträknade vinkeln från DSP:n till kameran utan att göra ytterligare beräkningar. Då blev kameran aldrig helt stillastående. Även i ett helt tyst rum, ett ekoisolerat rum, blev kameran aldrig helt stillastående vid en enkel talare/ljudkälla. Istället pekade den i ungefärligt rätt riktning med en hoppig variation upp mot $\pm 30^\circ$. När sedan en annan talare började tala, kanske på andra sidan av BigBroRoto, så rörde sig kameran inte mot den nya positionen i en jämt och fin rörelse utan snarare

hackade sig fram. Detta beror på medelvärdesbildningen i DSP:n.

Filtreringen i Raspberry Pi:n förbättrade stabiliteten i kamerans rörelse markant. Istället för att peka i ungefärligt rätt riktning och variera sig inom ett intervall blev kameran nu istället stillastående fram tills det att det skedde en uppdatering av vinkelns 5 gånger i rad som var större än $\pm 15^\circ$. Denna filtrering förbättrade också hackigheten som berodde på DSP:ns medelvärdesbildning. Dock fick den snabba uppdateringen uppföras som ledde till den lilla födröjningen som uppstod på ungefär 1 sekund.

4.2 Användarupplevelse

Hemsidan fungerade mycket bra och ger projektet överlag en mycket snygg och professionellt intryck.



Figur 4: FrontEnd - hemsidan med livefeed.

4.3 Flera ljudkällor

En fråga som är mycket intressant som berör detta projekt är vad som skulle hänta då det inte bara fanns en ljudkälla. Rent fysikaliskt rör sig ljudet fram 'konsonantvis' då en person talar och beroende på samplingshastighet kommer det bli 'luckor' mellan konsonanterna. Om algoritmen hade fungerat helt perfekt skulle DSP:n på grund av alla dess 'luckor' hinna räkna ut vinkelns för varje talare/ljudkälla då sannolikheten att deras konsonanter kommer till mikrofonerna exakt samtidigt är relativt liten. En sådan snabb uppdatering (dvs att kameran skulle ändra riktning mellan konsonanter i ett ord) skulle dock ej vara önskvärt av uppenbarliga skäl. I praktiken innebär flera ljudkällor en förvirrad algoritm där den mest högljudda ljudkällan (amplitud- och energimässigt) skulle få mest kameratid.

5 Återblick

Efter att projektet avslutats kunde det konstateras att målet med projektet, att bygga ett system som riktade en kamera mot en ljudkälla, hade uppnåtts. Genom hela projektets gång var den en god stämning inom gruppen och tidplanen hölls i stort sett genom hela projektet.

Eftersom BigBroRoto bestod av flera olika system som var tvungna att kommunicera med varandra med olika protokoll så fick mer tid än väntat läggas på att implementera dessa protokoll på de olika plattformarna. Bland annat fick SPI-protokollet implementeras på både DSP:n och Raspberry Pi:n för att dessa skulle kunna kommunicera med varandra.

En lärdom som drogs under projektet var att det hade varit en fördel om alla definitioner av koordinatsystem hade gjorts innan hård- och mjukvarugruppen började jobba. När systemet testades visade det sig att DSP:n hade en annan orientering för positiva vinklar än vad kameran hade. Detta ledde till att kameran inte riktades mot ljudkällan. Problemet åtgärdades men det visar att kommunikationen hade brustit inom gruppen. I övrigt fungerade kommunikationen mellan de olika arbetsgrupperna bra.

I projektets slutskede när signalbehandlingsalgoritmen skulle implementeras i C-kod på DSP:n visade det sig att DSP:n endast räknade ut vinklar i intervallet $[0, \pi]$. Det rådde osäkerhet om felet berodde på fel i implementationen i C eller om den teoretiska modellen inte stämde. Efter att ha gjort om alla beräkningar visade det sig att den teoretiska modell som användes inte var optimal. Efter att ha implementerat den nya modellen så fungerade allt som det skulle och systemet fick en dynamik i intervallet $[0, 2\pi]$. Detta ledde till några dagars försening men i projektplanen fanns det vissa marginaler för sådana här oförutsägbara händelser och projektet blev trots detta missöde färdigt i tid.

Ambitionsnivån på alla inblandade genom hela projektet var mycket hög. Detta gagnade projektet oerhört mycket. Men ambitionsnivån ledde även till lite komplikationer, gruppen hade lätt att låta idéerna bli allt för storskaliga och det krävdes mycket disciplin för att inte ta på sig en för stor arbetsbörd. Stor del av framgången inom projektet uppnåddes tack vare vår fantastiska handledare som låt ner tid för möten varje vecka och som alltid fanns till hands då det behövdes. En dedikerad handledare gynnade verkligen vårt projekt och nu i efterhand har vi svårt att se ett färdigt projekt utan Mikaela hjälp.

6 Slutsatser och fortsatt arbete

BigBroRoto var ett mycket lärrikt projekt som innehöll många olika moment i många olika kategorier. Alla medlemmar förbättrade sina egenskaper i grupparbete och även i att arbeta med projekt, vilket är mycket viktiga egenskaper för vår framtid som civilingenjörer. Det finns helt klart ett användningsområde för BigBroRoto vid till exempel videokonferenser eller vid övervakning. Projektgruppen är mycket nöjd med resultatet och arbetsinsatsen med BigBroRoto och ser detta som en merit för framtiden.

6.1 Viktning

Hade det funnits mer tid i detta projekt hade det nästa naturliga steget varit att i algoritmen introducera viktning. Man kan, genom att att implementera viktning reducera inverkan från ekon. Genom att testa BigBroRoto i ett ekoisolerat rum så kunde en bild fås om hur BigBroRoto hade fungerat om ekoreducerande viktning hade implementerats

6.2 Sökning i höjdled

För att få en mer användbar produkt hade det varit viktigt att implementera sökning i höjdled, det vill säga att kameran skulle riktas mot ansiktet på personen som talar. Detta implementeras inte i detta projektet eftersom det i stort sett hade varit samma algoritm som för sökning i sidled och därmed inte skulle gett någon väsentlig ny förståelse för signalbehandling. Dessutom hade en implementation av sökning i höjdled krävt att ett mer omfattande kommunikationsprotokoll så att Raspberry Pin skulle förstå om DSPn skickade en vinkel i höjd- eller sidled. Detta ansågs ligga utanför detta projekt.

Referenser

- [1] Swartling, Mikeal, *Direction of Arrival Estimation and Localization of Multiple Speech Sources in Enclosed Environments*, Blekinge Tekniska Högskola, 2012

A Appendix

A.1 Mötesprotokoll

19/2

Mötet den 19/2 påbörjades med att analysera den valda projektplanen tillsammans med handledaren för att få feedback och kritik på om tidsplanen verkade rimlig eller inte. Den handlade till mestadels dock om att komma igång med dem två olika huvuddelarna som projektet hade delats in i. Så båda grupperna hade en genomgång med handledaren om hur arbetet skulle påbörjas. Dem som skulle börja jobba med algoritmen fick en introduktion till hur simuleringen i matlab skulle se ut och hårdvarugruppen fick en liten genomgång om hur den valda DSP:n fungerade.

Till nästa möte sattes målen upp att algoritmgruppen ska så smått börja på matlabsimuleringen och hårdvarugruppen ska titta på framework-koden och läsa igenom processormanualen för DSP:n.

26/2

Till mötet den 26/2 har båda grupperna kommit igång med sina olika delar och genomgången av DSP:n och algoritmen med handledaren fortsatte utefter detta. Det konstaterades även att en Axis-kamera med dess utvecklingsmiljö skulle användas istället för andra jämförbara alternativ då detta ansågs smidigast. Denna ska sen styras av en Raspberry Pi via HTTP.

Till nästa möte ska jobbet med simuleringen av algoritmen fortsätta och PCB-layouten för DSP:n och dess utvecklingsmiljö VisualDSP++ med drivrutiner skulle undersökas.

12/3

Mötet den 12/3 handlade till stor del om att statusuppdatera om var alla låg till i projektet och hur vi skulle jobba vidare utifrån detta. Men även en liten genomgång av DSP:ns SPI gjordes med handledaren för lite bättre insikt i hur dem olika delarna ska kopplas ihop.

Till nästa möte ska arbetet fortsätta med uppgifterna som redan var etablerade sen förra mötet. Hårdvarugruppen ska utöver detta också börja titta på SPI-pinnarna och komma fram till vad som skulle vara bästa sättet att koppla samman DSPn med Raspberry Pi:n.

9/4

Till mötet den 9/4 var matlabsimuleringen av algoritmen klar och nästa steg att översätta denna till C-kod konstaterades för att kunna köra den på DSP:n. Det diskuterades också om vilken typ av mikrofoner som skulle användas och vilka som skulle vara mest optimala för vårt syfte. Det diskuterades också om möjligheterna att implementera ett front-end via en webbserver för att kunna se vad kameran riktar in sig mot med några enklare inställningar att styra kameran

nära till hands.

Till nästa möte ska arbetet fortgå som vanligt med dem redan givna uppgifterna och mikrofonerna ska beställas. Arbetet med webbservern ska också påbörjas så mycket som möjligt utifrån hur mycket tid som finns över.

16/4

Mötet den 16/4 handlade mycket om den statusrapport som ska lämnas in den 17/4. Så det diskuterades mycket om var vi förhöll oss jämfört med tidsplanen som tidigare hade satts upp i början av projektet. Även hur vi utifrån detta skulle jobba vidare med tidsmålen och hur den närmaste tiden kommer se ut.

Till nästa möte bestämdes att alla komponenter ska vara beställda och att programmeringen av Raspberry-Pi:n för att kommunicera med kameran ska vara klart. Annars ska arbetet fortgå som vanligt.

23/4

Till mötet den 23/4 ansågs C-kod implementeringen av algoritmen, front-end med webserver och programmeringen av Raspberry-Pi:n klart. Alla komponenter hade levererats och satts ihop till den modell som vi hade arbetat fram.

Så till nästa möte ska alla delar av projektet sättas ihop och alla problem som möjligent dyker upp under detta ska försöka lösas på bästa sätt.

11/5

Till mötet den 11/5 var algoritmen och hårdvaruimplementeringen klar och det fanns i stora drag ett fungerande system med mindre fel. Dessa undersöktes med hjälp av handledaren på mötet och lyckades tillslut lösa dessa på plats.

Fram till redovisningen ska systemet kalibreras marginellt och det ska konstrueras någon typ av platform för mikrofonerna så dem höjs och kan uppta ljud på ett så optimalt sätt som möjligt. Ställningen ska också försöka göras så att den andra hårdvaran inte kommer vara i vägen för dessa mikrofoner.

18/5

På det extrainsatta mötet den 18/5 lades en effektspärr till i systemet på plats för att BigBroRoto ska röra sig mer kontrollerat under redovisningen. Utöver detta blev mötet en sista avstämning innan presentationen med reflektioner över vad som hade gjorts bra och mindre bra under projektets gång.

A.2 GIT-repo

Koden som skrevs i projektet versionshanterades med Git. All kod som gjordes under projektets gång kan ses på github.

<https://github.com/alexandernajafi/bigbroroto>

A.3 Huvudkod i DSP:n

Följande kod är kärnan av koden i DSP:n, huvudalgoritmen. Dock är det bara en liten del av den sammanlagda koden.

```
void algorithm()
{
    int i;

    /*
    Disable interrupts , load data from codecs
    */

    interrupt(SIG_SP1,SIG_IGN);
    sample_t *u32 = dsp_get_audio_01();
    sample_t *u33 = dsp_get_audio_23();

    for(i = 0; i < N; i++)
    {
        x1[i] = (float) u32[i].left;
        x2[i] = (float) u32[i].right;
        x3[i] = (float) u33[i].left;
        x4[i] = (float) u33[i].right;
    }

    int n;
    for(n = 0; n < halfN; n++)
    {

        GA[n] = cmltf(X1[n], conj(X2[n]));
        GB[n] = cmltf(X2[n], conj(X4[n]));
        GC[n] = cmltf(X4[n], conj(X3[n]));
        GD[n] = cmltf(X3[n], conj(X1[n]));
    }

    for(i = 0; i < halfN; i++)
    {
        GAmean[i] = caddf(GA[i], GAmean[i]);
        GBmean[i] = caddf(GB[i], GBmean[i]);
        GCmean[i] = caddf(GC[i], GCmean[i]);
        GDmean[i] = caddf(GD[i], GDmean[i]);
    }

    /*
    Calculate steered response power
    */
}

packageCounter++;
if(packageCounter -1 == nbrOfPackages)
{
    int k;
    for (k = 0; k< halfN; k++)
    {
        GAmean[k].re = GAmean[k].re/nbrOfPackages;
```

```

GAmean[k].im = GAmean[k].im/nbrOfPackages;
GBmean[k].re = GBmean[k].re/nbrOfPackages;
GBmean[k].im = GBmean[k].im/nbrOfPackages;
GCmean[k].re = GCmean[k].re/nbrOfPackages;
GCmean[k].im = GCmean[k].im/nbrOfPackages;
GDmean[k].re = GDmean[k].re/nbrOfPackages;
GDmean[k].im = GDmean[k].im/nbrOfPackages;
}

complex_float exponentCos1;
exponentCos1.re = 0;
complex_float exponentSin1;
complex_float exponentCos2;
exponentCos2.re = 0;
complex_float exponentSin2;
exponentSin2.re = 0; ;
exponentSin1.re = 0;
complex_float sumTot;

int j;
float maxAngle = 0;
float maxVal = 0;
float tmpAngle;
complex_float amlt;
complex_float bmlt;
complex_float cmlt;
complex_float dmlt;
complex_float inputSin1;
complex_float inputCos1;
complex_float inputSin2;
complex_float inputCos2;

for(j = 0; j<resolution; j++)
{
    tmpAngle = angles[j];
    sumTot.re = 0;
    sumTot.im = 0;
    for(i = 9; i <lenTmpVec; i++)
    {

        exponentCos1.im = -1 * cosf(tmpAngle) *
            angularFreq[i] * conversionConstant;
        exponentSin1.im = -1 * sinf(tmpAngle) *
            angularFreq[i] * conversionConstant;
        exponentCos2.im = 1 * cosf(tmpAngle) *
            angularFreq[i] * conversionConstant;
        exponentSin2.im = 1 * sinf(tmpAngle) *
            angularFreq[i] * conversionConstant;

        inputSin1 = cexpf(exponentSin1);
        inputCos1 = cexpf(exponentCos1);
        inputSin2 = cexpf(exponentSin2);
        inputCos2 = cexpf(exponentCos2);

        amlt = cmltf(GAmean[i], inputCos1);
        bmlt = cmltf(GBmean[i], inputSin1);
        cmlt = cmltf(GCmean[i], inputCos2);
        dmlt = cmltf(GDmean[i], inputSin2);

        sumTot = caddf(sumTot, amlt);
        sumTot = caddf(sumTot, bmlt);
    }
}

```

```

        sumTot = caddf(sumTot, cmlt);
        sumTot = caddf(sumTot, dmlt);

    if(sumTot.re >= maxVal)
    {
        maxVal = sumTot.re;
        maxAngle = tmpAngle;
    }
}

/*
Send calculated angle with SPI
*/

if(maxVal > 100)
{
float trans = 255/360.0*180/pi*maxAngle;
spi_send((int)trans);
prevAngle = maxAngle;
}

packageCounter = 1;

/*
Reset mean values after SRP algorithm
*/

for (k = 0; k < halfN; k++)
{
    GAmean[k].re = 0;
    GBmean[k].re = 0;
    GCmean[k].re = 0;
    GDmean[k].re = 0;
    GAmean[k].im = 0;
    GBmean[k].im = 0;
    GCmean[k].im = 0;
    GDmean[k].im = 0;
}
}

void main()
{
    dsp_init();
    dsp_start();
    test();
    algorithm_setup();
    interrupt(SIG_SP1,algorithm);
    while(1)
    {
        interrupt(SIG_SP1,algorithm);
        idle();
    }
    algorithm_close();
}

```