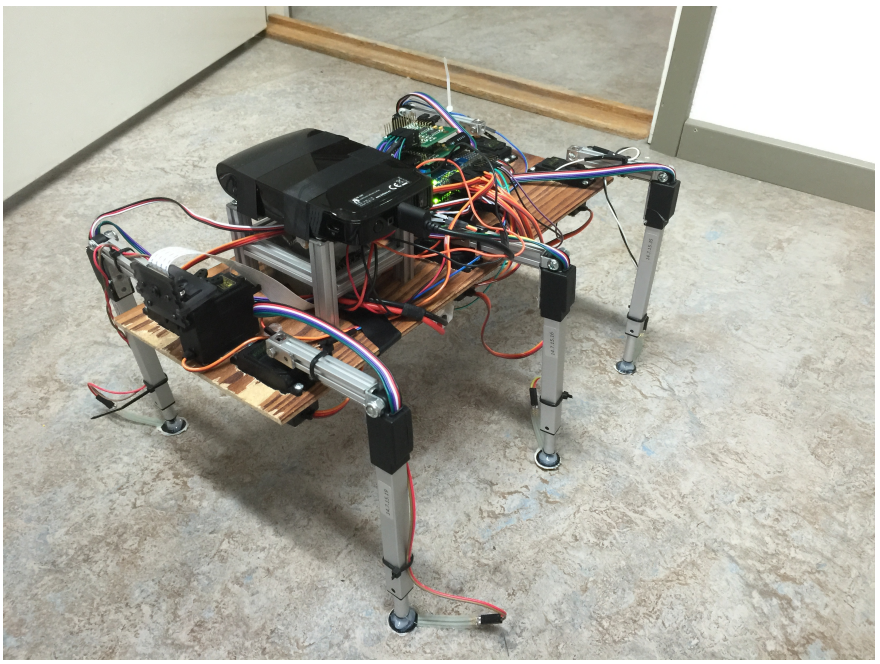
CHALMERS



# Gudrun - fjärrstyrt terränggående fordon
# Gudrun - remote controlled terrain vehicle

SSYX05


Högskoleingenjörprogrammet Mekatronik

Isabelle Nilsson

Simon Modigh

Examinator Bertil Thomas

Institutionen för signaler och system

CHALMERS TEKNISKA HÖGSKOLA

Göteborg, Sverige, 2015

# Förord

För vår del är detta examensarbetet den avslutande kursen på Chalmers Mekatronikutbildning 180 hp som omfattar 15 hp.
Vi har fått möjligheten att göra vårt examensarbete på Broccoli Engineering AB där vi vill tacka Björn Bergholm som kom på denna intressanta och spännande idéen.

Vi vill också tacka -
vår handledare på Chalmers, Göran Hult
vår handledare på Broccoli Engineering, Martin Skogsberg
samt vår examinator, Bertil Thomas

Hoppas på trevlig läsning!
Isabelle Nilsson och Simon Modigh

# Sammanfattning

Efter orkanen Gudrun uppstod flera problem för alla inblandade. Bland annat var det problem med att ta sig fram i de berörda områdena. Ett smidigt sätt för att färdas i svår terräng fanns inte tillgängligt, därför utvecklades detta projekt för att försöka konstruera en miniatyr prototyp av ett terränggående fordon som klarar av ojämnheter samt hinder i svår terräng.

De som skulle kunna vara intresserade av detta fordon kan vara de som vill utforska områdena, t.ex. räddningstjänst, försäkringsbolag, markägare, organisationer som Missing People Sweden etc.

Resultatet har blivit en fjärrstyrd robot med sex ben som med hjälp av cylindrar kan ta sig över ojämn terräng genom att med tryckkänsliga resistorer känna av när de når sitt underlag. Denna robot har fått namnet Gudrun, döpt efter stormen Gudrun. För att röra sig framåt använder den sig av sex servon som har potential att röra sig ungefär 180°. Roboten styrs av en förare med hjälp av en handkontroll som kommunicerar med en Raspberry Pi, en enkortsdator, via bluetooth och den i sin tur ger ut en video stream via WiFi som kan visas på en dator i samma nätverk.

I det fall som en fullskalig version skulle byggas kan denna prototyp vara en bra riktlinje för hur man skall gå till väga, även om en större version troligen skulle kunna vara av bättre kvalitet och med bättre stabilitet. Tidsbegränsningen och den begränsade kunskapen om hur komponenterna skulle fungera har gjort att prototypen blivit lite instabil samt har vissa funktioner blivit uteslutna.

Detta arbete har utförts på Broccoli Engineering AB i Lundby, Göteborg.

# Summary

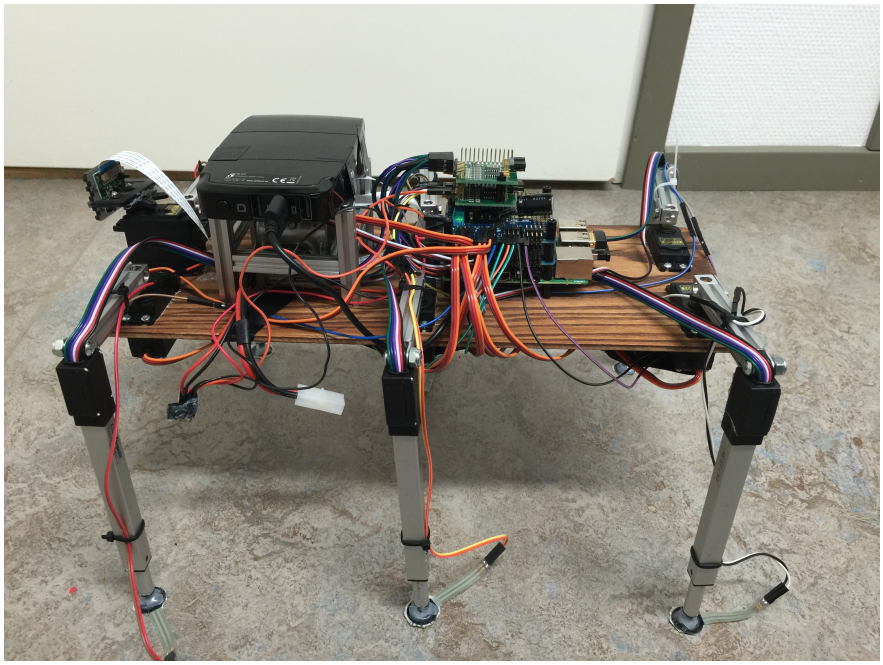
After the cyclone Gudrun (Erwin) a number of problems occurred for everyone involved. Among other things there were problems in accessing the affected areas of the forrest. A efficient way to maneuver in tough terrain wasn't available, therefore this project was developed to construct a miniature prototype of a all terrain vehicle that handles uneven terrain and obstacles in tough terrain conditions.

The interested parties of this project would be the organisations that wants to examine damages e.g. emergency personnel, insurance companies, landowners, Missing People Sweden etc.

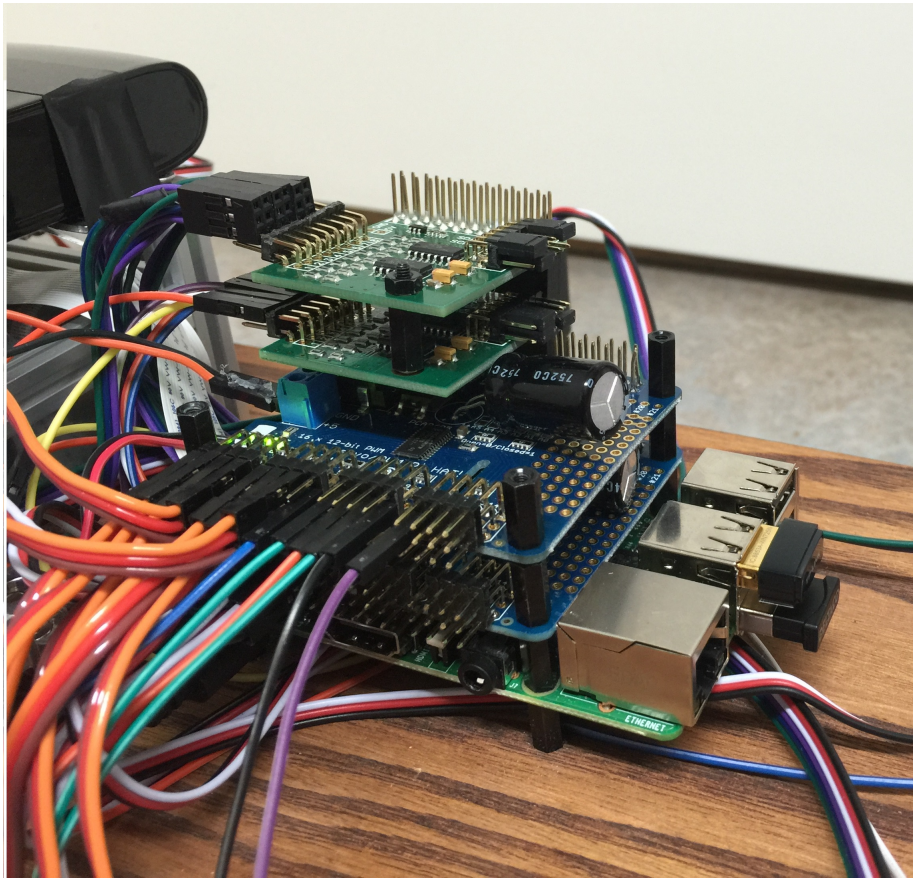
The result is a six legged robot that with the help of cylinders can manouver through tough terrain and by using forcesensitive resistors feel when the legs have reached the surface. To move forward it uses six servos that have the potential to move approximatly 180°. The robot is controlled by a operator with a control that communicates with a Raspberry Pi through bluetooth that in turn sends a WiFi videostream that can be viewed by a computer in the same network. The prototype is wireless.

In the case a fullscaled version would be constructed this prototype would be a good guideline for how to construct it, even if a larger model probably would be able to be of better quality and with better stability. The timelimit and the limited knowledge about how the components would work have made the prototype a little bit unstable aswell as it forced many features to be eliminated.

The project has been executed at Broccoli Engineering AB in Lundby, Gothenburg.



Övergripande bild på roboten (*Författarens bild*)



Kretskopplingen (*Författarens bild*)

# Innehåll

# Beteckningar

**PWM** - Pulse-width modulation.
**I$^2$C** - Inter-Integrated Circuit, en synkron seriell multimasterbuss.
**IDE** - Integrated Development Environment (programmeringsmiljön).
**GPIO** - General-purpose input/output (pins).
**RC** - Radio Control (Radiostyrning).
**CPU** - Central Processing Unit (Processor).
**RAM** - Random Access Memory (arbetsminne).
**HDMI** - High-Definition Multimedia Interface, för övervföring av video och ljud
**GPU** - Graphics processing unit (grafikprocessor)
**ADC** - Analog to Digital Converter (Analog till Digital konverterare)
**FPS** - Frames Per Second (bilder per sekund)
**WiFi** - Trådlöst nätverk

# 1 Inledning

## 1.1 Bakgrund

Björn Bergholm, VD på Broccoli Engineering AB, hade under en tid funderat på hur man skulle kunna ta sig fram i skog efter en storm. Denna fundering uppstod efter att han sett en dokumentär om stormen Gudrun som drabbade stora delar av Sverige i Januari 2005. Björn kom på idéen att problemet kunde lösas med ett fordon som hade teleskopisa ben. Vilket skulle möjliggöra att ta sig an samtliga hinder. Ur denna idé kom intresset att se hur en verklig prototyp skulle utvecklas.

## 1.2 Syfte

Syften med detta arbete är att konstruera ett fordon som har teleskopiska ben för att kunna röra sig i ojämn terräng så som t.ex. skog efter storm mm. Målet är att ha en fjärrstyrd maskin som kan ta sig fram i svår terräng. Viktiga delmål är att ta fram en prototyp för fordonet i mindre skala. Därefter programmera prototypen så den uppfyller kravet på rörlighet. Sista steget är optimering av fordonet.

## 1.3 Avgränsningar

Fordonprototypens krav är att kunna gå i ojämn terräng med en begränsning på höjden för hinder. Det finns det inga krav på räckvidd eller på hur vädertålig produkten skall vara. Den skall kunna styras manuellt och behöver inte vara autonom, samt finns inga krav på att kunna bära last. Ekonomiska hänsynstagande tas i den utsträckning att roboten håller sig i en nedskalad version av tänkt produkt, vilket leder till kostnadsminskning av komponenter. Utifrån denna skala så görs inköp med avseende på prestanda och krav, alltså vilka komponenter som är mest optimala för prototypen.

Den behöver inte -

- klara väta eftersom det är en tidig prototyp och den då är tänkt att uteslutande testas inomhus.
- komma över allt för höga hinder, hur höga hinder den kan ta sig över är direkt kopplat till hur långa de teleskopiska benen är. P.g.a. av de små dimensionerna på fordonet så finns inte ett optimalt utbud vilket leder till att det inte är rimligt med för höga krav på detta.
- helautomatiseras. En idé hade kunnat vara att göra roboten helt automatisk för att utforska skogen men detta hade varit väldigt tidskrävande och väljes därför bort.
- kunna ta sig upp för branta backar då detta hade lett till en mycket större arbetsuppgift.
- ha lång räckvidd. Vi kommer använda Bluetooth teknik för att detta är mer känt av författarna och tänker utgå från dess räckvidd då den som tidigare nämnts bara ska köras inomhus.
- hålla någon viss hastighet.

- ha videofeed. För att på stora avstånd kunna styra den utan att ha överblick hade en ”videofeed” varit en möjlig funktion. Då tid finns är detta dock en bra funktion för att öka prototypens funktionalitet.

## 1.4 Precisering av frågeställningen

- Kommer robotens design klara av de krav som ställs?
- Vad är lämplig styrning av roboten?
- Kommer roboten kunna byggas i verklig skala för att sedan sättas i bruk för det arbete som den är tänkt att klara av?
- Finns det en bra lösning för att klara av hinder?
- Är Raspberry Pi en bra lösning?

# 2 Teknisk bakgrund

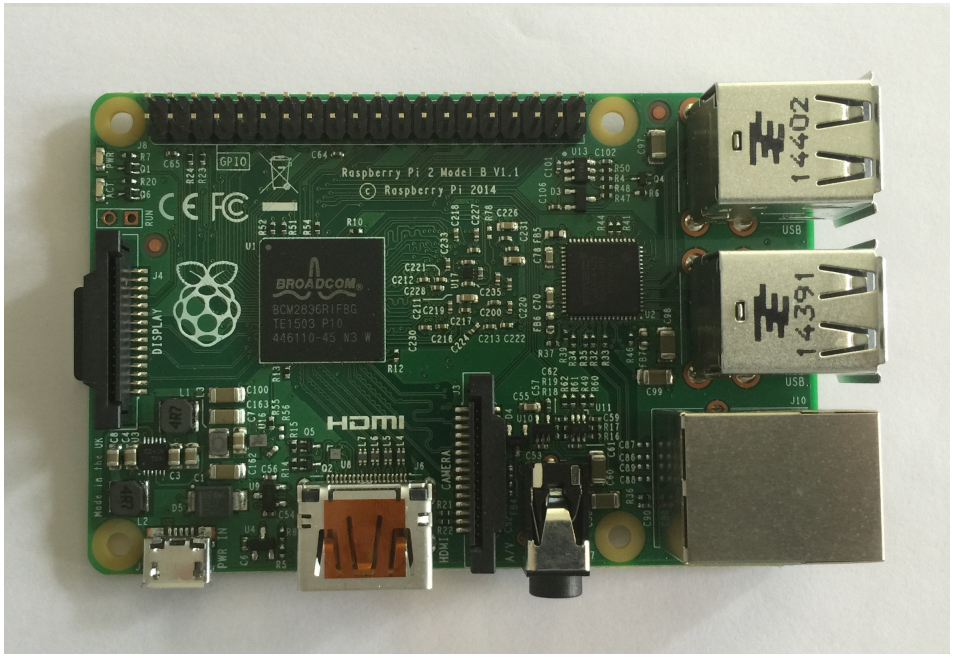
## 2.1 Raspberry Pi

Raspberry Pi är en enkortsdator utvecklad av Raspberry Pi Foundation. Raspberry Pi Foundation är en välgörenhetsföretag baserat i Storbritannien som har som mål att lära både barn och vuxna mer om datorer och programmering. Skaparna av Raspberry Pi var baserade på *University of Cambridge's Computer Laboratory* och de ville ge bättre dator- och programmeringskunskaper hos de nyintagna som hade sämre kunskaper för varje år. [1]

Raspberry Pi finns i flera modeller, några med 26st GPIO (Modell A/A+) och andra med 40st (Modell B/B+). Den nyaste modellen kallas Raspberry Pi 2 (Modell B) som har högre prestanda än sina föregångare, närmare bestämt en 900 MH*z quad-core ARMCortex-A7* som CPU, 1Gb ramminne, och en *VideoCore IV 3D graphics core* som GPU. [2]

Den har också 40 GPIO, HDMI utgång, 4 USB ports, ethernetport, kamerainterface, displayinterface samt kombinerad 3.5 mm ljudkontakt och kompositvideo. [2]

Raspberry Pin används ofta inom hobbyprojekt då för att den relativt billigt, har många användningsområden och trots dess storlek är kraftfull nog att användas som surfdatorer.[3]



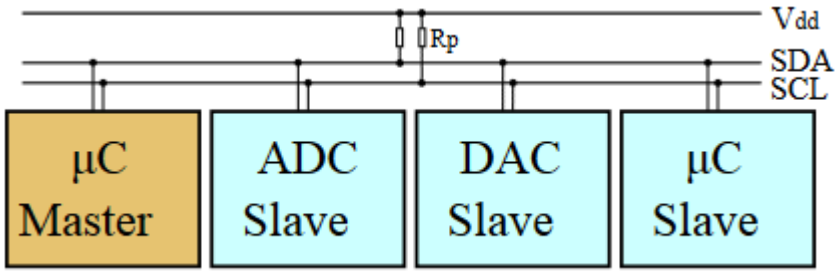

Figur 1: Raspberry Pi 2 model B (*Författarens bild*)

## 2.2 I²C

I²C (också kallat I2C eller IIC) står för "Inter-Integrated Circuit" och är en synkron seriell multimasterbuss som utvecklats av Philips under tidigt 80-tal. Den utvecklades för att kunna kommunicera mellan en CPU och TV-apparater. Nuförtiden används den även av de flesta konkurrenterna i bl.a. mobiltelefoner och datorers moderkort. Sedan 2006 har det varit licensfritt att använda I²C, men det finns fortfarande en avgift för att erhålla slavadresser från förtaget NXP (tidigare Philips Semiconductors). [4] [5]

Intel utvecklade 1995 "SMBus" som baserades på I²C, den utvecklades för att ha större kompatibilitets möjligheter och bättre stabilitet och används därför oftare i moderna system än Philips föregångare, även om båda protokollen ibland stöds. [6]

I²C använder två ledare, en för datasignalen (SDA - Serial Data Line) och en för klocksignalen (SCL - Serial Clock), för tvåvägskommunikation. Standarden för inspänningen är +3.3 V eller +5 V. En sak att ha i åtanke är att bussen inte kan ha obegränsad längd utan bussens kapacitans måste vara under 400 pF. [4]




Figur 2: I2C kommunikation, [7]

## 2.3 Python

Python är ett högnivåprogrammeringsspråk som gavs ut tidigt 90-tal som klarar flera sorters programmeringsparadigmer så som objektbaserad-, imperativ-, funtionell- och procedurell programmering. Den finns i flera olika versioner, den nyaste i dagsläget är Python 3 (ibland kallat Python 3000 eller py3k). Python är det språk som rekommenderas att användas med Raspberry Pi och det finns ett flertal färdigskrivna exempel och bibliotek i Python. [8] [9] [10]

Python sätter högt värde på läsbarhet, att det är viktigt att lätt kunna läsa koden och förstå vad som utförs. Denna syntax leder i vissa fall till att det behövs en mindre mängd kod för ett program i python jämfört med motsvarande program i t.ex. C programmering. [11]

I Python så behöver man inte deklarera variabler så som i t.ex. C, utan den deklarerar variabeltyp själv utifrån hur den används.

En av de mest utmärkande skillnaderna är att istället för användning av t.ex. { } eller [ ] för att innesluta satser så använder sig python av indentering.

Som exempel så ser en if sats med print ut i python som följande:

```
if True:
    print "Hello World!"
```

samma programkod men skrivet i C:

```
if(True)
{
    printf("Hello World!");
}
```

I C koden finns en indentering precis som i python fast skillnaden är att den inte fyller någon funktion, utan skrivs enbart för att öka läsbarheten.

# 3 Metod

Innan projektstarten så utvecklades en preliminär planeringsrapport och en grundläggande plan för hur prototypen skulle utvecklas. När projektet började så var första steget att välja ut de komponenter som skulle användas, i och med detta var det nödvändigt att lite mer ingående bestämma hur prototypen skulle fungera och röra sig. Under första veckan skickades beställningar på majoriteten av de komponenter som krävdes och en Raspberry Pi inköptes i butik för att slippa leveranstiden. Tiden som blev över medans leveranserna inväntades spenderades på installation att lära sig hantera IDEn Geany och experimentera med GPIOs samt installation av själva operativsystemet på raspberryn. Första programmet som utvecklades var blinkande lampor som sedan omvandlades till att kontrollera servon när de levererades.

Sedan gällde det att börja försöka bygga upp ett fungerande program för rörelse. Till att börja med var det ganska svårt att se om det fungerade i praktiken på grund av att prototypen inte var komplett och ihopsatt än. En stor del var också att kunna styra prototypen med hjälp av en handkontroll, Sonys Dualshock 3, via bluetooth. Med stor hjälp från diverse guider och forum fick vi det till slut att fungera.

När cylindrarna levererades var det dags att montera ihop prototypen. Basen till roboten var en träplatta som modifierades för att kunna fästa raspberryn och servona. I servona fästes sedan en distans för att få cylindrarna en bit ut från plattan. Distansen var från början gjord i trä men byttes sedan ut till metall för att öka stabiliteten.

När den var färdig monterad och klar testades programmet, och modifikationer kunde sedan implementeras för att t.ex. rätta tankefel i programmeringen.

Då grunden för gången var byggd testades körning av cylindrarna innan de fästes på roboten. Sedan programmerades cylindrarna beroende av det ben då fästes på.

När allting såg bra ut var det dags att testa det trådlöst, alltså utan att använda nätaggregaten samt trådlöst nätverk.

Det sista som utfördes var att implementera att cylindrarna skulle känna av om de nått marken. Detta gjordes genom att koppla 5 V från en PWM hatt till tryckkänsliga resistorer (en på varje ben) och sen vidare till en ADC hatt som konverterade signalen så att vi kunde använda den i programmeringen. Utöver detta hade vi ytterligare en ADC hatt för att hantera positions feedbacken ifrån cylindrarna.

Under projektets gång lades extrafunktioner till då tid fanns.

# 4 Problemanalys

Nedbrytning av projektmål i delmål

- Teleskopsben
  Pris, slaglängd, storlek, vikt, kraft, motering, placering, fastsättning, typ av drivning, frihetsgrader.
  Antal 4, 6 eller 8 ben?

- Programmering
  Benens algoritm, Hur skall den röra sig framåt? Hur skall den svänga?
  Rörelse för benen vid gång.
  Hur får man benen att röra sig i rätt gradtal? Hur vet benen när de nått marken?

- Motorer
  Pris, kraft, storlek, vikt, enkelhet, montering, placering, fastsättning, koppling.

- Batteri
  Kapacitet, storlek, säkerhet, pris, enkelhet, montering, koppling.

- Kretskort
  Användarvänlighet, programmeringsspråk, pris, versatilt, tillbehör, montering, fastsättning, koppling.

- Fjärrstyrning/Bluetooth

  **Hårdvara**
  Tillbehör till kretskort? "Dongel" – Separat Bluetoothenhet? Vilken fjärr-/handkontroll?

  **Mjukvara**
  Kommunikation mellan Bluetooth enheten och kretskortet, kommunikation mellan kontrollen och enheten, programmera knapparna på kontrollen till rätt funktion.

- Vikt
  Får inte väga mer än benens kapacitet, ha en bra tyngdpunkt för tillräcklig balans.

- Nivåreglering
  Automatisera höjdregleringen för att ta sig över hinder.

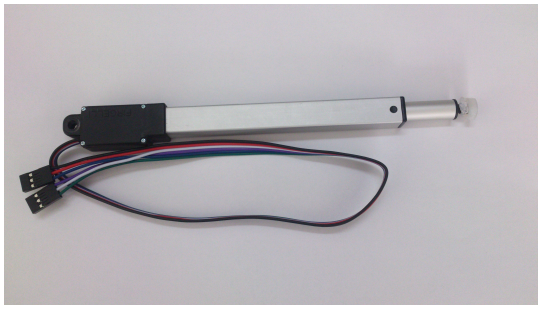
# 5 Val av komponeneter

## 5.1 Linjär aktuator

De linjära aktuatorerna, även kallat ställdon, är till för att kunna justera benens längd under färd för att anpassa roboten till den varierande marknivån.

Den valda aktutatorn är Firgelli L12 6 V med 50:1 utväxling, 100 mm slaglängd och styralternativ ”I”. [12]

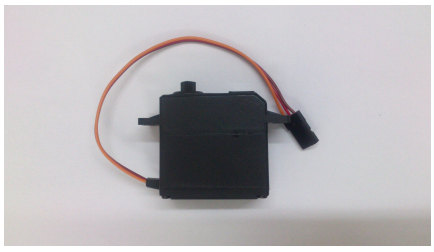
De valdes för att storleken, vikten (56 g) och styrkan (tillsammans klara lyfta ca 7 kg) var väldigt bra med tanke på uppgiften de har samt att priset var lägre än många av de andra alternativen. Att styrsystem ”I” valdes var för att de hade möjligheten att styras med hjälp av varierande ström, varierande spänning, RC input eller PWM signal, samt att de har en analog feedbak som visar hur långt ut aktuatorn har gått.[13] Det innebar flest variationer för styrning av cylindrarna, vilket är bra då det kan uppkomma komplikationer man ej tänkt på under processen samt bra för vidareutveckling.



Figur 3: Ställdon (*Författarens bild*)

## 5.2 Servomotorer

Servomotorerna som valdes var Tower-Pro MG995. Sett till vad de beräknades klara av duger de servona då de kommer utsättas för väldigt lite kraft. Deras storlek talade också för att dessa servon valde (40x20x40 mm) och det faktum att det var metall växellåda. Dessa servon var inte kontinuerliga utan hårdvarulåsta på ungefär 180 grader. Vikten var låg (55 g) vilket är bra då ställdonen får det lättare att arbeta. [14]



Figur 4: Servo (*Författarens bild*)

## 5.3 Dualshock 3 SixAxis

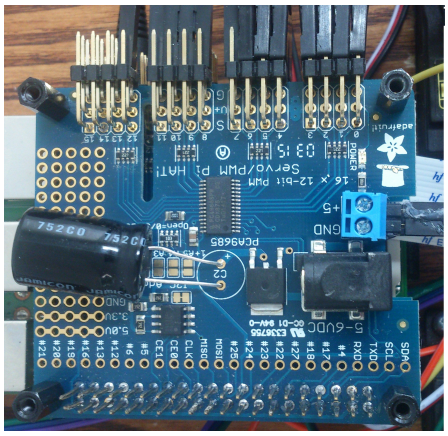
Dualshock 3 SixAxis är en bluetooth handkontroll tillverkad av Sony för användning till spelkonsolen Sony PlayStation 3 från 2006. Denna handkontroll används ofta i hobby projekt med Raspberry Pi och därför finns det gott om dokumentation. Därav valdes denna handkontroll då valet av kretskort föll på Raspberry Pi. Handkontrollen finns i några olika versioner, den nyaste ”SixAxis” har också ”Rumble” funktion vilket original utgåvan inte hade. Är bra att ha denna kunskap då många dokument hänvisar till denna Rumblefunktionen. [15]

## 5.4 PWM hatt

För styrning av servon och cylindrar krävs 12-13 stycken PWM-signaler, Raspberry Pi 2 modell B har tillräckligt med utgångar men det krävs mjukvarumässig PWM styrning på utgångarna förutom på två stycken channels som har hårdvarumässig PWM.[16] Frekvensen är begränsad samt krävs egen strömförsörjning till varje ben för att inte överbelasta Raspberry Pin.[17] Därav används en ”Adafruit 16-Channel PWM/Servo HAT för Raspberry Pi”. PWM-hatten har hårdvarumässig PWM signal, den är anpassad till servokontakter och information om PWM-hatten finns lättillgänglig på internet.[18]

Servona har en pulstid på 20 ms vilken ger en frekvens på 40 Hz.[14] Ställdonen fungerar ej stabilt under 100 Hz (prövning gav detta resultat då information om Ställdonens duty cycle ej hittades). PWM-hatten har 16 kanaler och kan alltså styra 16 olika motorer samtidigt men dock enbart med en frekvens per PWM hatt.[19] Därav används två stycken PWM-hattar för att att kunna styra både cylinder och servo.

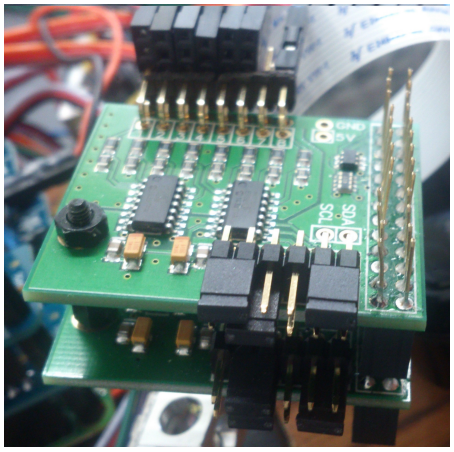
Det går också att ”stacka”, eller stapla, upp till 62 stycken Adafruit 16-Channel PWM/-Servo Hattar (vilket alltså ger upp till 992 separata PWM signaler). Detta är möjligt genom adressändring. Adressändring sker hårdvarumässigt genom att löda ihop olika adress på PWM-kortets ”jumpers” beroende på vilken adress du vill ha (basadressen ligger på 0x40) [18]



Figur 5: (*Författarens bild*)

## 5.5 ADC Pi

ADC Pi (V.2) är ett påbyggnadskort till Raspberry Pi från AB Electronics UK. Den är en analog till digital konverterare som kommunicerar med Raspberry Pi via I2C och har ett färdigt python 3 bibliotek att ladda ner.
ADC Pi används för att ta emot den analoga signalen som de tryckkänsliga sensorerna avger. Signalen konverteras till digital signal så Raspberry Pin kan behandla dess data. Den kan även användas för att ta emot den analoga feedbacken från ställdonen. [21]



Figur 6: (*Författarens bild*)

## 5.6 Bluetooth adapter

För kommunikation mellan Raspberry Pi och handkontroller behövs det en bluetooth adapter. Den valda adaptern ”Deltaco Bluetooth v4.0 nano adapter” använder bluetooth version 4.0. Dess räckvidd ligger på 10 meter vilket räcker för syftet med den fjärrstyrda roboten. Den har USB 2.0 vilket är kompatibelt med Raspberry Pi:n. [22]

## 5.7 Kamera modul

Pi camera module är en kamera utvecklad av Raspberry för Raspberry Pi. Den använder den speciella kontakten för Camera Interface som finns på Raspberry Pin.
Då roboten är fjärrstyrd underlättar en kamerastyrning vid längre räckvidd samtidigt ökar också robotens användningsområden. [23]

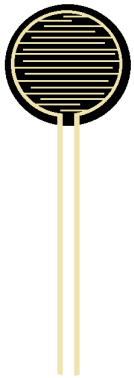
## 5.8 Tryckkänsligt motstånd

För att klara av hinder som t.ex. en upphöjning behövs någon typ av avkänning för att Raspberryn ska få information om detta. Med denna information ska då komponenter tillsammans med Raspberry:n behärska detta hinder och ta sig förbi det. Lösningen på problemet blev en resistor som ger olika resistans beroende på tryck. Vid 0 $N$ ligger resistansen på 1 $M\Omega$, mindre resistans ju mer tryck. Den kopplas in via 2.54 $mm$ stiftanslutning och ger ut en analog signal. [24] Signalen konverteras till digital signal (se 5.5

ADC Pi) och därav får Raspberry:n information om hindret och ger tillbaka signalen att den har nått sitt mål.

```
- Mätområde 100g-10kg
- Tryckytans diameter: 12,7mm (0.5")
```

[24]



Figur 7: Tryckresistor (*Författarens bild*)

## 5.9 WiFi adapter

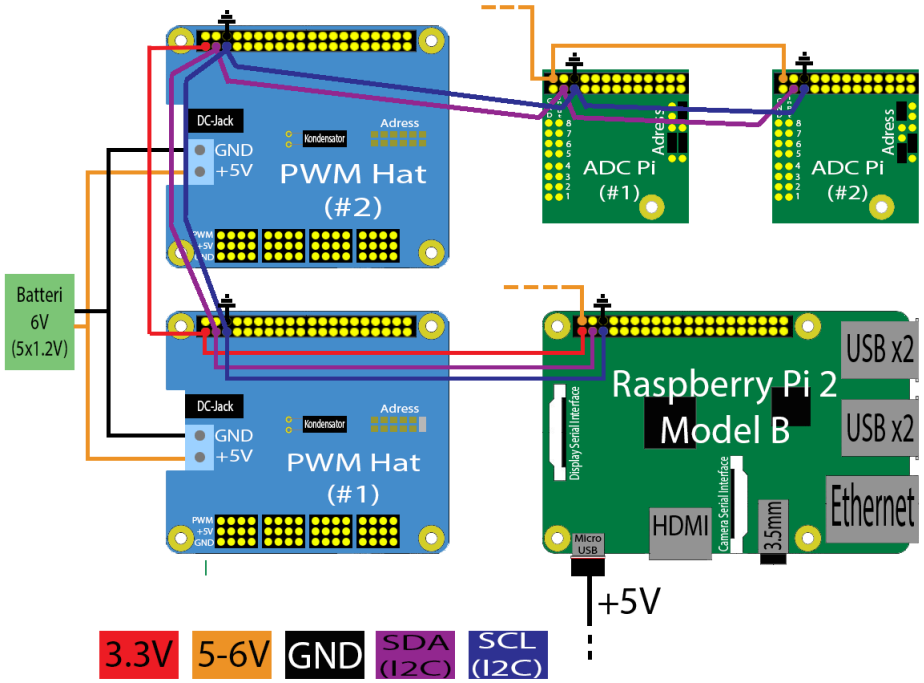
För att göra prototypen helt trådlös och kunna streama video trådlöst köptes en WiFi adapter, ”TP-Link TL-WN725N”. Den valdes för att den var kompatibel med raspberry pi och klarar av kontorets WiFi. [25]

## 5.10 Raspberry Pi

Anledningen att just denna enkortsdatorn valdes var p.g.a. prestanda då processorkraften är 900 MHz och den har en fyrkärnig processor samt 1 GB RAM-minne, möjligheten att välja mellan flera olika programmeringspråk, anslutningsmöjligheter b.la. Wifi, bluetooth etc., tillbehör och tillgänglig information.[3]

# 6 Konstruktion

## 6.1 Koppling




Figur 8: Grundläggande koppling (Författarens bild)

Uppkopplingen av påbyggnadskorten görs genom att ”stacka” dem på varandra, och eftersom ADC Pin har färre pins så kopplas den in på de första 26 pinsen (vänster från bilden sett, Figur 2).
(För mer information om GPIO se bilaga A.8)

Även om alla pins är inkopplade används bara en liten del av dem, medan resten av dem bara kopplas vidare till våningen över. Det som alla korten har gemensamt är att de utnyttjar $I^2C$ för att kommunicera med Raspberry Pi:n. Detta görs via pin 2 för SDA och pin 5 för SCL, i bilden ovan (Figur 1) avbildat med lila respektive blå kabel. För att detta ska fungera med flera olika kort samtidigt behöver de olika adresser. Detta appliceras genom att löda ihop ”jumpers” på PWM-hattarna och genom att koppla in ”jumpers” på ADC Pi:ns Adress pins. Vilket visas i figur 1, ADC Pi 1 och 2 Adress. ADC Pin använder 2 adresser per kort för att det är dubbla AD Converter pic’s på dem. [18] [21]

Start adressen för PWM-hattarna är 0x40 och genom lödning på ena kortets adress ”jumpers” kan adressen ändras till t.ex. 0x41. Adresserna för ADC Pi korten har valts till 0x68 och 0x69 samt 0x6A och 0x6B.

Förutom $I^2C$ kablarna behövs strömförsörning. Raspberryn behöver 5 V, med minst 1 A, fast kan behöva upp till 2,1 A om det är många strömkrävande funktioner och tillbehör kopplade till den. PWM-hattarna behöver 3,3 V som ges från pin 1 på Raspberryn, detta

för PWM-signalen. De behöver även 5-6 V från en extern strömkälla så som nätaggregat eller batteri för strömförsörjning till komponenter som kopplas in på kortet. Hur mycket ström den kräver är beroende av hur många och hur strömkrävande komponenter är som kopplas in, för användning av alla kanaler rekommenderas mellan 8-10 A.[18] ADC Pin kräver 5 V från pin 2 på raspberryn som kopplas via, fast används inte av, PWM-hattarna.

Sedan krävs givetvis att jord kopplas in på alla korten, t.ex. till pin 6 på raspberryn (se bilaga A.8). Ground pin ligger också på pin 6 på alla korten.

## 6.2 Montering

Grunden till roboten är en träplatta. I denna träplatta sågades hål ut för att servona skulle kunna läggas i dem och sedan skruvades servona fast i botten.

Hål borras i bottenplattan för att kunna fästa Raspberry Pi:n samt batteriramen.

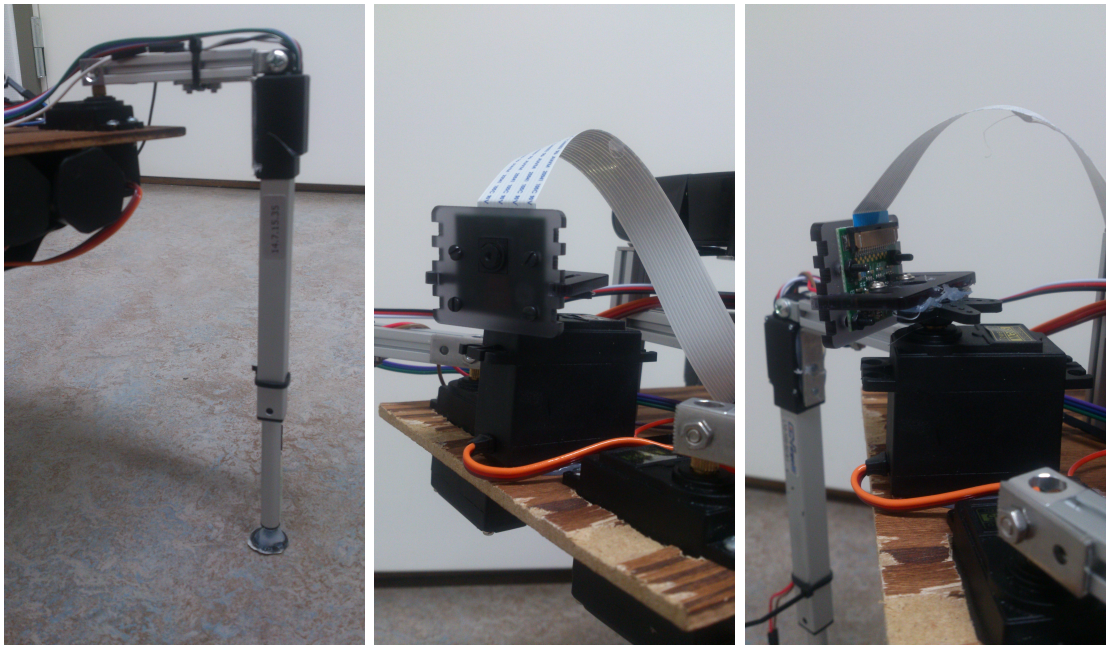
Bendistanserna är gjorda i aluminium och är av märket Makerbeam, gjorda för protyping. Med hjälp av tillbehöret ”Corner cubes” kan man skruva fast ena ändan av balkarna i servona och sedan skruva fast cylindrana i andra ändarna med hjälp av medföljande infästning. Ett vinkeljärn (90°) limmas fast i balken och cylinderna för att garantera stabilitet. [26]

Tryckkänsliga resistorer limmas fast vid cylindrarnas ände. Eftersom greppet mot marken blir dåligt limmas också sandpapper fast på resistorernas undersida.

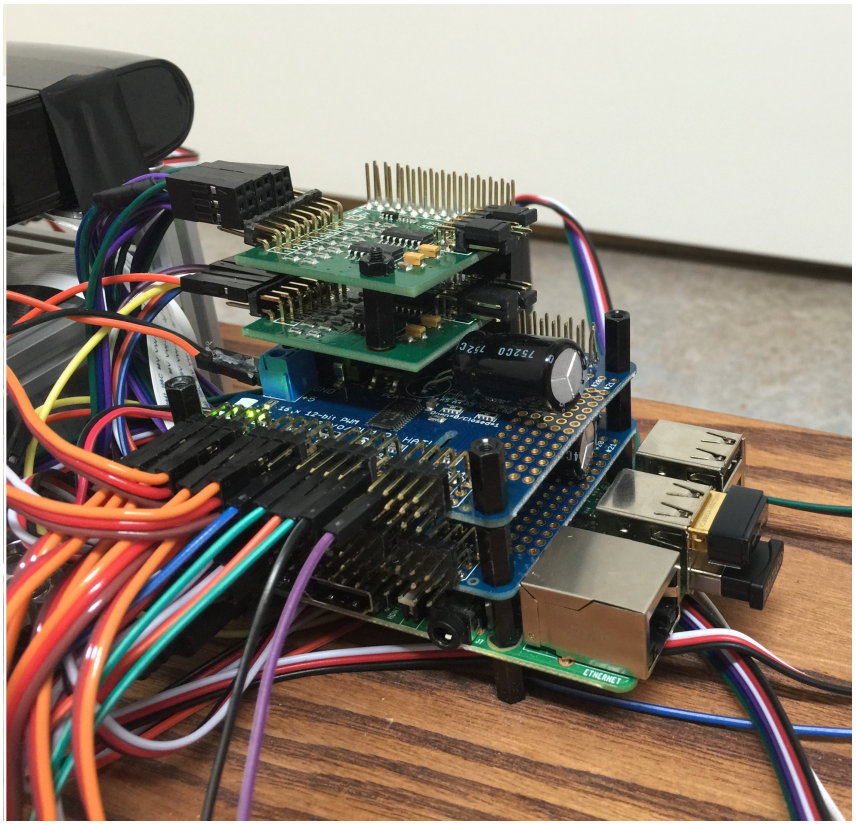
Längst fram på roboten limmas ett servo fast för att kunna fästa kameran på den. För att underlätta användingen av den används en ”Camera Mount - Pimoroni”.

På Raspberry Pin monteras PWM-hattarna och på dem ”stackades” även ADC Pi korten.

På batterihållaren tejpas ”powerbanken” fast och i batteriramen läggs 6 V batteriet.



Figur 9: Konstruktion av ben samt kameran fäst på servo för rotation (*Författarens bild*)



Figur 10: Kretskorten stackadepå varandra (*Författarens bild*)

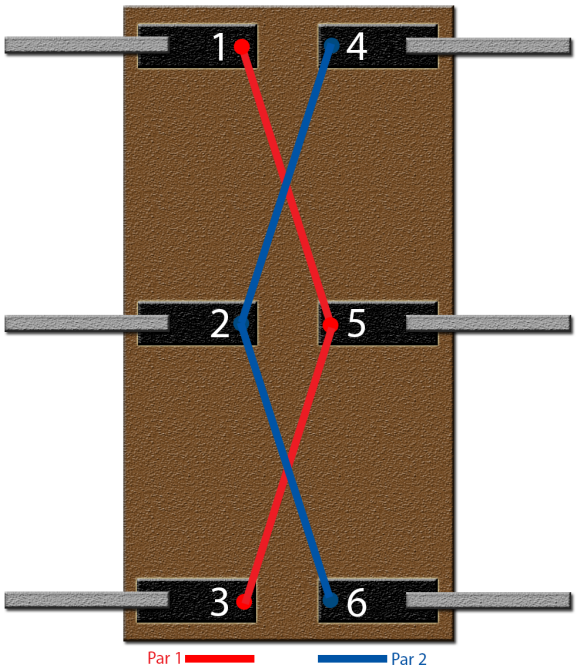
## 6.3 Struktur för gång

För att få roboten att gå används idèen av en spindels gång som grund för konstruktionen. Spindelns ben jobbar i par, två ben på högersidan jobbar ihop med två ben på vänstersidan. När två ben lyfts på högersidan, lyfts även två ben på vänstersidan, resterande ben har kontakt med marken. Benet på andra sidan gör inversen av det som sker på motsvarande sida. Om spindeln lyfter på höger ben fram så lyfts ej vänstra benet fram, om vänster ben rör sig framåt rör sig högerbenet bakåt.

Skillnaden som blir mellan prototypen och en spindels gång är att prototypen enbart har 6 ben och en spindel har som känt 8. Därav är prototypen inte så stabil som det hade önskats.

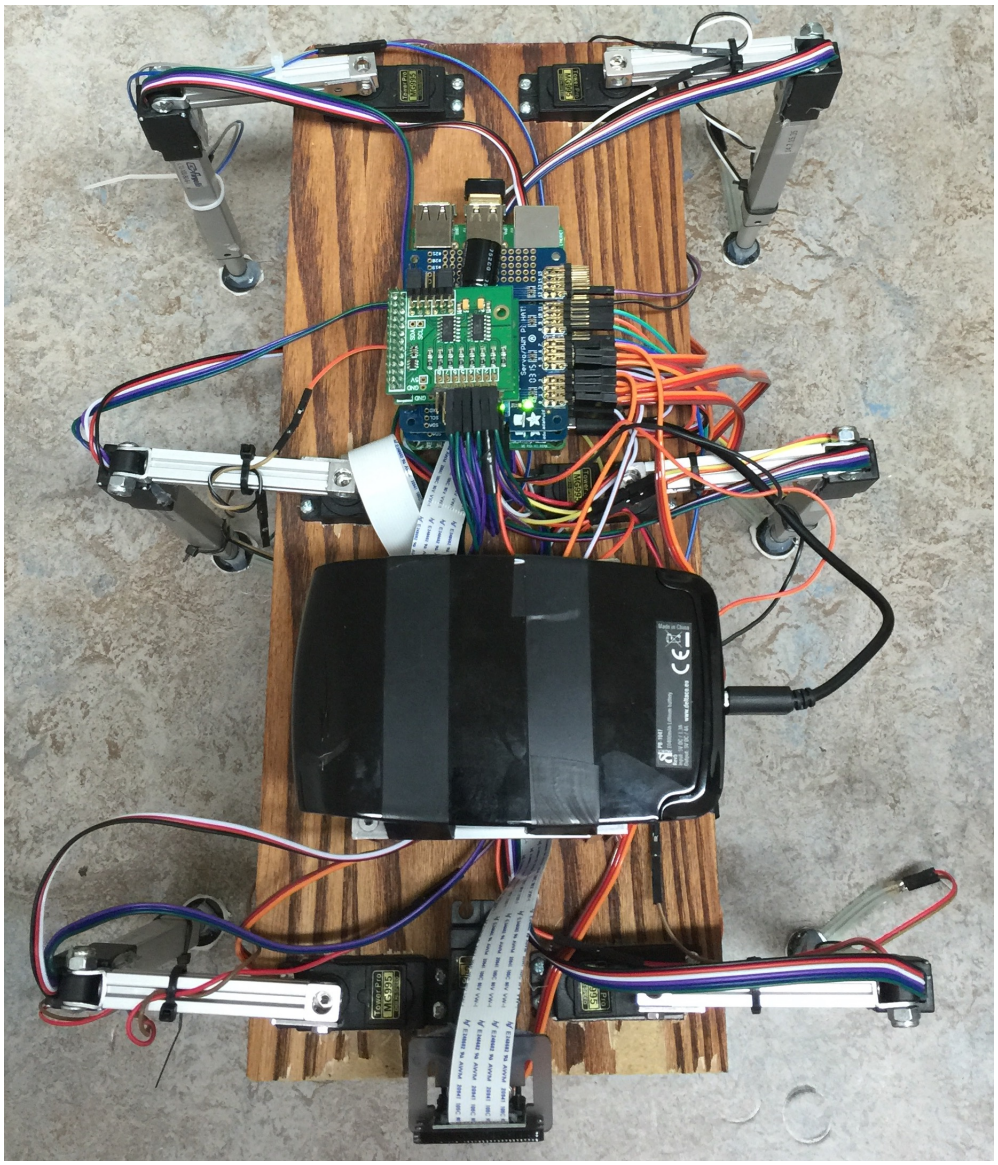
Benen (då menat servo, distans och cylinder) jobbar i två par. Ben 1, 3 och 5 (par 1), samt ben 2, 4 och 6 (par 2) har samma rörelse algoritm dock är det en fördröjning emellan dessa båda paren. Om par 1 lyfts stannar par 2 kvar och har kontakt med marken tills dess att par 1 åter får kontakt med marken, då har fördröjningenförflutit och par 2 lyft. Det som sker under fördröjningen"är att de par som lyfts ska röra sig framåt medans de par som fortfarande har kontakt med marken ska röra sig bakåt, detta är då roboten rör sig (åt önskat håll). Sedan inväntas att cylindrarna får kontakt med marken och signal skickas till Raspbery:n innan motsvarande par rör sig uppåt.

Roboten har även en funktion som svänger (Turn() i bilaga A.8). Det som sker då är att en sidan går kortare steg i jämförelse med motsvarande sida. Detta kan justeras i koden beroende hur kraftigt man vill svänga.




Figur 11: (*Författarens bild*)

# 7 Resultat



Figur 12: Gudrun (Författarens bild)

Prototypen som konstruerades var en robot med sex ben, som kan höjas 10 cm, med ett tryckkänsligt motstånd på varje ben. Varje ben är fäst på ett servo som kan vridas ungefär 50-60°. Längst fram på roboten sitter en kamera fäst på ett servo som kan roteras ungefär 180°, kameran kan streama video till en dator på samma WiFi. Roboten styrs av ett Raspberry Pi kort som med hjälp av två PWM-hattar och två ADC Pi kort kontrollerar rörelsen. Föraren styr vad roboten skall göra med hjälp av en handkontroll som kommunicerar med Raspberry Pi:n via bluetooth. Raspberry Pin får sin ström via en ”powerbank” som sitter monterad på roboten och som ger ut 5 V, och PWM-hattarna får från ett separat 6 V batteri. Då de ska strömförsörja servon och cylindrar.

# 8 Slutsatser

Syftet av uppgiften var till att börja med inte särskilt specificerat, den skulle ha teleskopiska ben och kunna gå i ojämn terräng. Utöver detta fanns ingen plan för hur prototypen skulle fungera eller vilka funktioner den skulle ha. Prototypen kan på avstånd styras och klarar av hinder som ej överstiger en viss höjd eller ett visst djup (max 10 cm, se 5.1). Prototypen kan även utvecklas till viss mån, detta var ett önskat krav.

**Kommer robotens design klara av de krav som ställs?**
Till viss del, ett problem som uppstått är att tyngdpunkten inte är centrerad vilket leder till att de tryckkänsliga resistorerna får olika mycket tryck även om det är samma förhållanden. Förutom detta håller designen de krav som ställts på funktion.

**Vad är lämplig styrning av roboten?**
För prototypen är en simpel handkontroll ett bra val, då det går att styra utan att behöva dra kablar till styrningen som kan trasslas in. Vid en uppskalad version eller en version som skall ha längre räckvidd än 10 m (se WiFi 5.9) kan styrning via internet vara ett bättre alternativ.

Slutsatsen är att för de ändamål som prototypen har är det ett bra val med en vanlig handkontroll.

**Kommer roboten kunna byggas i verklig skala för att sedan sättas i bruk för det arbete som den är tänkt att klara av?**
En uppskalad version av denna prototyp ska gå att konstruera. Det som behövs är längre och kraftfullare cylindrar, kraftfullare servon och batterier samt tåligare sensorer för avkänning. Raspberry Pi:n bör fortfarande gå att använda som kretskort men ytterligare komponenter får läggas till då större strömmar och spänningar ska hanteras.

**Finns det en bra lösning för att klara av hinder?**
För att klara av den terräng som prototypen stöter på skall den kunna veta hur långt benen skall sänkas. Detta löstes genom tryckkänsliga motstånd under varje ben. Alternativt skulle detta gå att lösa genom avståndssensorer på varje ben och dessa skulle då kunna bestämma hur långt benen skulle gå ner. Problemet med detta system skulle ha varit att det inte skulle gå att mäta precis där benet skulle hamna då den skulle mäta lite brevid och dessutom att det skulle vara mindre pålitligt och dyrare.

Slutsatsen som fattades av detta var att det var det bästa alternativet att använda tryckkänsliga resistorer då den dessutom kräver en viss kraft innan den registerar och därför fungerar bättre när terrängen är mjuk, istället för en avståndsmätare som inte hade tagit hänsyn till det.

**Är Raspberry Pi 2 en bra lösning?**
Som alternativ hade man kunnat använda Arduino, Banana Pi eller kopplat upp ett eget system. Banana Pi är en mycket likande produkt fast med bättre prestanda och lite högre pris, med en nackdel som är att det finns mindre dokumentation och support till den.

Dessutom finns det större utbud av tillbehör och mjukvara till Raspberry Pin. Detta gör att för personer som inte har stor erfarenhet inom ämnet har lättare att lära sig Raspberry Pi:n än Banana Pi. Arduino är till skillnad från Raspberry Pi och Banana Pi inte enkorts datorer utan en micro-controller, den har sämre prestanda än dessa, men eftersom den inte har ett operativsystem så påverkar det inte lika mycket. Den har också färdiga analoga ingångar som man kan koppla in sensorer etc. utan att behöva AD omvandla. Nackdelarna är att trådlös kommunikation skulle blivit svårare att genomföra då den inte är tillverkad för det. En egentillverkat micro-controller hade antagligen gett bättre resultat men hade tagit mycket mer tid att genomföra och hade troligen inte kunnat slutföras under projektets tidbegränsning. [27] [28]

Slutsatsen som dras av detta är att om man inte har erfarenhet av liknande projekt innan så rekommenderas att man använder sig av en Raspberry Pi 2 Model B. För optimalt resultat hade det troligen varit bäst att kombinera en Raspberry Pi och en Arduino, då de går att koppla ihop via t.ex. usb, för att få fördelarna av båda men inte drabbas av nackdelarna.

# 9 Diskussion

## 9.1 Produktens goda egenskaper

Prototypen Gudrun är fjärrstyrd och kan ta sig fram till viss del bland ojämn terräng. Den är enkel att modifiera då den är relativt liten, vilket gör att komponenterna är överkomliga i pris. Det går att lägga till / ta bort saker ganska så enkelt då det är en öppen konstruktion. Prisnivån för komponenter i denna storlek gör också att det går att lägga till funktioner som gör att produkten bör stiga i värde mer än den stiger i kostnad. T.ex. kostar en kamera som är Raspberry Pi kompatibel 249 kr[29]. För denna kostand ökar dock produktens användningsområden och förmodligen gör det produkten mer tilltalande för andra intresserade.

## 9.2 Produktens mindre bra egenskaper

De tryckkänsliga resistorerna som ska se till att cylindrarna går till rätt längd är svåra att ställa in så att roboten håller sig plan. Raspberryn läser av spänningen som kommer in men som programmerare är det svårt att veta värdena vid de olika fallen som kan inträffa. Därav blir roboten ej stabil utan kan börja luta.

I dagsläge går det enbart att köra en funktion i taget, detta skulle kunna lösas genom att köra med trådar i programmet. Detta är inte något som drar ner så mycket på prototypens prestanda med den skulle bli snabbare, samt skulle problem som uppkommer kunna lösas fortare.

Vid gång används servon som rör sig inom ett visst gradtal en formel används för att försöka hålla koll på vart de befinner sig. Dock är denna formel inte optimal men fungerar så pass bra att de funktioner som finns idag kan utföras. Vid byte av vissa funktioner kan gången bli lite hackig men enbart i servots första rörelse. Detta skulle dock kunna avvärjas med hjälp av återkoppling så programmet bara kan kolla av vart de befinner sig istället för att matematiskt försöka hålla koll.

## 9.3 Missöden under projektets gång

Under de första veckorna då montering och testkörning av servona gjordes skedde vissa missöden. Vid första testkörningen av ett servo började PWM-hatten att ryka. Svaret på varför detta skedde är ännu oklart. Ett servo var inkopplat så överhettning verkar inte vara svaret. Misstankar om fabrikationsfel finns. En andra gång började PWM-hatten ryka, dock kördes fler servon denna gången och det började ryka då DC-kontakten kopplades in. Ingen kondensator hade lötts på PWM-kortet vilket kan vara en förklaring till denna överhettning, men detta förblir enbart spekulationer.

## 9.4 Idéer för vidareutveckling

Under projektets gång har flera idéer uppstått för hur man skulle kunna förbättra robotens egenskaper. Dessa idéer kunde inte genomföras pga tidsbrist men vid eventuell vidareutveckling skulle följande kunna genomföras.

- Gyro
  En funktion som hade kunnat förbättra rörelsen är om ett gyro hade installerats för att kontrollera lutningen av robotens bottenplatta, detta hade kunnat hjälpa till med att förhindra att roboten skulle välta.

- Servos med feedback
  Feedback skulle kunna hjälpa servona att röra sig effektivare.

- Radar
  Radar hade kunnat hjälpa till att upptäcka hinder i god tid så att de eventuellt skulle kunna undvikas.

- Mekanisk robotarm
  Kunna lyfta objekt ur vägen eller kunna bära med dem.

- Multithreading/multiprocessing
  Denna funktion skulle kunna göra programmet smidigare och snabbare.

- Bättre räckvidd
  Implementera styrning t.ex. via internet för att kunna styra från långt avstånd. Då skulle också ett gps chip eller kompass kunna hjälpa till.

- Förbättra videostreaming
  Streamingen är med låg FPS och ganska dålig kvalitet, detta skulle kunna förbättras betydligt. Dessutom skulle det vara bättre om den inte krävde trådlöst nätverk för att fungera.

- Styrning från mobiltelefon/surfplatta.
  En applikation skulle kunna utvecklas för alternativ styrning

- Allmän stabilisering/optimering
  T.ex. stabilare hårdvara, mindre/lättare batterier eller bättre prestanda.

- Kunna starta programmet utan inloggning på Raspberry Pin.
  Slippa köra operativsystemet parallellt med programmeringen samt slippa ha en skärm inkopplad för att kunna starta den.

# Referenser


[1] About us, The Making of Pi, Raspberry Pi Foundation
https://www.raspberrypi.org/about/
(Acc 2015-05-27)

[2] Raspberry Pi 2 Model B, Adafruit Industries/Raspberry Pi Foundation
http://www.adafruit.com/pdfs/raspberrypi2modelb.pdf
(Acc 2015-05-27)

[3] Raspberry Pi, Kjell och Company
http://www.kjell.com/fraga-kjell/teman/raspberry-pi
(Acc 2015-06-01)

[4] $I^2C$ Info, I2C Info, i2c.info.
http://i2c.info/
(Acc 2015-05-26)

[5] I2C History, Embedded System Academy
http://www.esacademy.com/en/library/technical-articles-and-documents/miscellaneous/i2c-bus/general-introduction/history-of-the-i2c-bus.html
(Acc 2015-06-01)

[6] SMBus facts, telos Systementwicklung GmbH
http://www.telos.info/p/hw/traciixl20/facts-smbus/
(Acc 2015-05-26)

[7] I2C Wikipedia
http://sv.wikipedia.org/wiki/I%$C2$%$B2C$
(Acc 2015-06-03)

[8] What is Python?, Python Software Foundation
https://www.python.org/doc/essays/blurb/
(Acc 2015-05-27)

[9] Python readme, Raspberry Pi Foundation
https://www.raspberrypi.org/documentation/usage/python/
(Acc 2015-05-26)

[10] Raspberry Pi FAQ, Raspberry Pi Foundation
https://www.raspberrypi.org/help/faqs/#softwareLanguages
(Acc 2015-05-29)

[11] The Making of Python, A Conversation with Guido van Rossum, by Bill Venners
http://www.artima.com/intv/pythonP.html
(Acc 2015-05-26)

[12] L12 Actuator Series, Firgelli Technologies Inc.
http://store.firgelli.com/category_s/1849.htm
(Acc 2015-05-27)

[13] Miniature Linear Motion Series L12, Firgelli Technologies Inc.
http://www.firgelli.com/Uploads/L12_datasheet.pdf
(Acc 2015-05-27)

[14] TowerPro MG995 basic information, Servo Database, servodatabase.com
www.servodatabase.com/servo/towerpro/mg995
(Acc 2015-05-27)

[15] DualShock 3 Wireless Controller, Sony
https://www.playstation.com/en-us/explore/accessories/dualshock-3/
(Acc 2015-06-01)

[16] Raspberry Pi 2 Hardware PWM, by rpdom
https://www.raspberrypi.org/forums/viewtopic.php?f=91&t=105044
(Acc 2015-06-01)

[17] Software PWM Library, Gordons Projects, @drogon
https://projects.drogon.net/raspberry-pi/wiringpi/software-pwm-library/
(Acc 2015-05-27)

[18] Adafruit 16-Channel PWM/Servo HAT for Raspberry Pi, Adafruit Industries
https://learn.adafruit.com/downloads/pdf/adafruit-16-channel-pwm-servo-hat-for-raspberry-pi.pdf
(Acc 2015-05-27)

[19] PCA9685 General discription, NXP Semiconductors
http://www.nxp.com/documents/data_sheet/PCA9685.pdf
(Acc 2015-05-27)

[20] Adafruit 16-Channel PWM/Servo HAT for Raspberry Pi, Adafruit Industries
https://learn.adafruit.com/adafruit-16-channel-pwm-servo-hat-for-raspberry-pi?view=all
(Acc 2015-05-27)

[21] ADC Pi introduction, AB Electronics UK
https://www.abelectronics.co.uk/products/3/Raspberry-Pi/17/ADC-Pi-V2—Raspberry-Pi-Analogue-to-Digital-converter
(Acc 2015-05-28)

[22] DELTACO Bluetooth nano-adapter, BT-118, SweDeltaco AB
https://www.deltaco.se/products/items/itemid/%28BT-118%29/index.aspx
(Acc 2015-05-27)

[23] Camera Module, Raspberry Pi Foundation
https://www.raspberrypi.org/products/camera-module/
(Acc 2015-05-26)

[24] Tryckkänsligt motstånd 0.5”, Electrokit Sweden AB, tillverkade av VersaPoint Technology
http://www.electrokit.com/tryckkansligt-motstand-0-5.46954
(Acc 2015-05-25)

[25] 150Mbps Wireless N Nano USB Adapter, TP-Link
http://www.tp-link.com/lk/products/details/?model=TL-WN725N
(Acc 2015-05-31)

[26] MAKERBEAM Products, Makerbeam
http://www.makerbeam.eu/epages/63128753.sf/en_GB/?ObjectPath=/Shops/63128753/Categori
(Acc 2015-05-30)

[27] Raspberry Pi vs. Banana Pi, by Brad Bourque
http://www.digitaltrends.com/computing/raspberry-pi-vs-banana-pi/
(Acc 2015-05-29)

[28] Arduino vs. Raspberry Pi, by Brad Bourque
http://www.digitaltrends.com/computing/arduino-vs-raspberry-pi/
(Acc 2015-05-29)

[29] Raspberry Pi Kamera
http://www.prylstaden.se/index.php?option=com_$virtuemart$&$Itemid$ =
68&$page$ = $shop.product\_details$&$flypage$ = $flypage.pbv.v1.tpl$&$product\_id$ =
1046&$category\_id$ = 210&$gclid$ = $CKTg297\_7sUCFaUNcwodzUUAPA$
(Acc 2015-05-01)

# A Bilagor

Vanliga kommandon som återkommer i dessa bilagor

**sudo** = kör kommando som super user
**wget** = hämta fil
**tar** = packa upp .tar filer
**cd** = gå til directory
**git** = Kommando för att arbeta med git
    **-clone** = Skapar en Git repository kopia från en källa
    **-rm** = tar bort filer från index
**apt-get** = Kommando för att arbeta med APT filer (Advanced Packaging Tool)
    **-update** = identifiera nya filer som finns att ladda ner
    **-upgrade** = ladda ner nya filer (måste köra update först)
    **-install** = installera nya filer
**mkdir** = Skapa ny mapp
**uname** = Skriv ut system information på skärmen

## A.1 Installation av Bluetooth dongeln

**All text skrivs in i terminalen**

sudo apt-get install bluez-utils bluez-compat bluez-hcidump
sudo checkinstall libusb-dev libbluetooth-dev joystick

**Para ihop Bluetooth dongeln med Ps3 kontroller**
Ladda ner och kompilera programmet till Raspberry:n:
sudo apt-get update
sudo apt-get install pyqt4-dev-tools
sudo wget http://www.pabr.org/sixlinux/sixpair.c
sudo gcc -o sixpair sixpair.c -lusb

Paring:
sudo ./sixpair

Joystick manager:
wget http://sourceforge.net/projects/qtsixa/files/QtSixA%201.5.1/QtSixA-1.5.1-src.tar.gz
tar xfvz QtSixA-1.5.1-src.tar.gz
cd QtSixA-1.5.1/sixad
make
sudo mkdir -p /var/lib/sixad/profiles
sudo apt-get install checkinstall
sudo checkinstall
sudo sixad –start

Gör så att programmet för att hantera handkontrollern alltid starta vid uppstart av

Raspberry:n:
sudo update-rc.d sixad defaults

omstart krävs
reboot

Denna guide hämtades ifrån http://www.raspians.com/Knowledgebase/ps3-dualshock-controller-install-on-the-raspberry-pi/ 03-2015

## A.2 Wifi adapter

Installation av wifi adapter (model nummer TL-WN725N) på Raspberry:n
Uppdatera Raspberry Pi
rpi-update

(sudo apt-get install rpi-update - om föregående ej funkar skriv in detta kommado först)

uname -a (för att få fram operativsystems versionsnummer)

sudo apt-get update (uppgradera - identifiera förändringar av systempaketet och uppdatera)
sudo apt-get upgrade
sudo reboot

**Installera drivrutiner**
Denna installation är beroende av vilken version raspberry pi:n har. Därav byts "8188eu-20140307öch sökvägarna i exemplet nedan ut, beroende av version (för att veta operativsystem version skriv "uname -a"i terminalen)

Fil att ladda ner hittas på Raspberry pi forumet:
$http://www.raspberrypi.org/forum/viewtopic.php?p = 462982$

wget https://dl.dropboxusercontent.com/u/80256631/8188eu-20140307.tar.gz
tar -zxvf 8188eu-20140307.tar.gz
sudo install -p -m 644 8188eu.ko /lib/modules/3.10.33+/kernel/drivers/net/wireless
sudo insmod /lib/modules/3.10.33+/kernel/drivers/net/wireless/8188eu.ko
sudo depmod -a
sudo reboot

## A.3 Konfiguration av PWM HAT

**Enable I2C**

sudo nano /etc/modules
Lägg till följande två rader längst ner
i2c-bcm2708

i2c-dev

sudo nano /etc/modprobe.d/raspi-blacklist.conf
Kommentera bort följande rader (sätt # framför)
blacklist spi-bcm2708
blacklist i2c-bcm2708

sudo nano /boot/config.txt
Lägg till följande 2 rader
dtparam=i2c1=on
dtparam=i2c_arm=on

sudo i2cdetect -y 1
För att testa, två I2C adresses skall synas

**Python bibliotek**
sudo git clone https://github.com/adafruit/Adafruit-Motor-HAT-Python-Library.git
sudo cd Adafruit-Motor-HAT-Python-Library
sudo apt-get install python-dev
sudo python setup.py install

sudo cd examples
Exempel koder för att testa servos

## A.4 GPIO Library (Python)

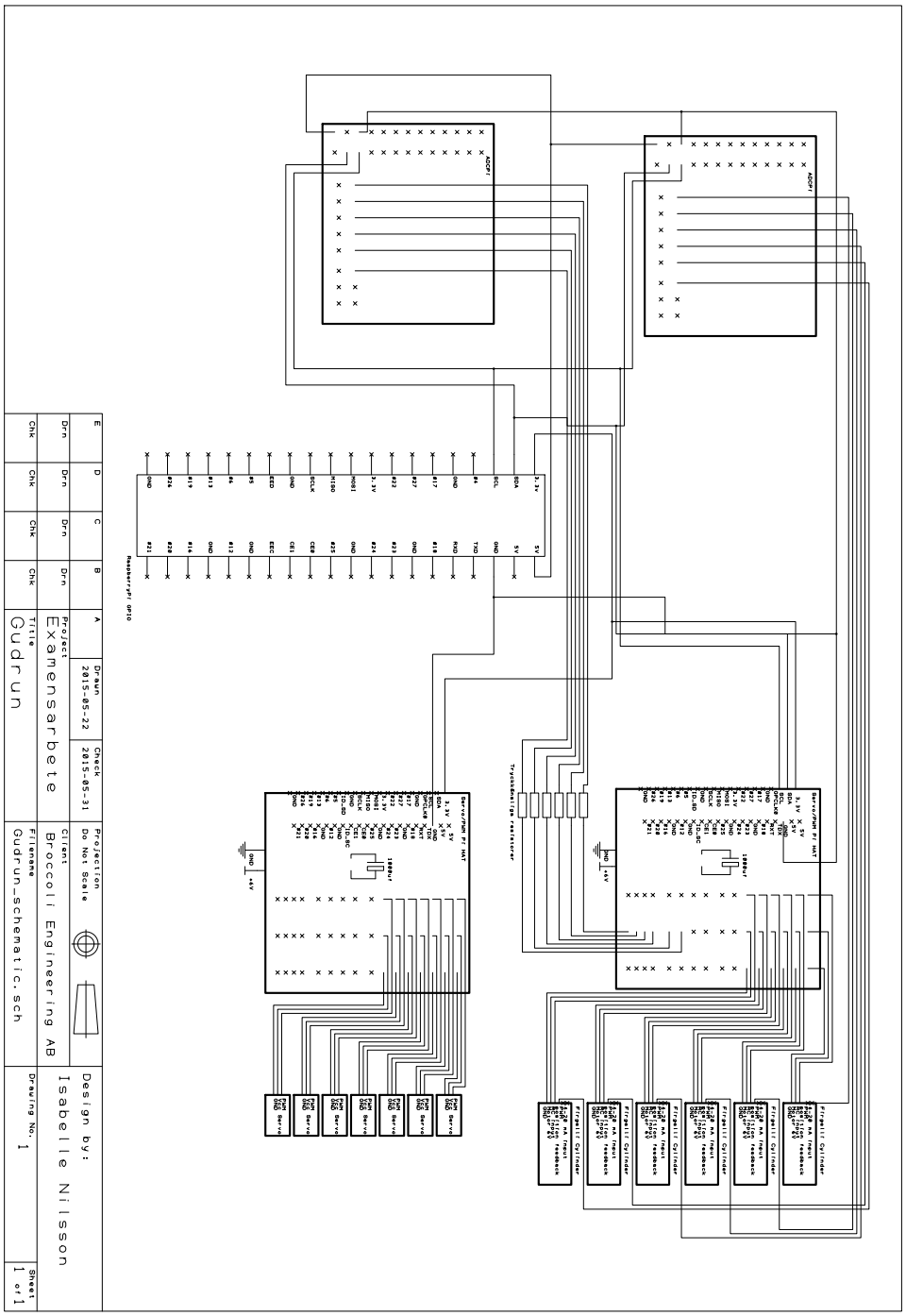
PyPa - rekommenderat verktyg för att installera Python-paket.
curl -O https://raw.github.com/pypa/pip/master/contrib/get-pip.py
sudo python get-pip.py

(bibliotek)
sudo wget http://pypi.python.org/packages/source/R/RPi.GPIO/RPi.GPIO-0.1.0.tar.gz
sudo tar zxf RPi.GPIO-0.1.0.tar.gz
sudo cd RPi.GPIO-0.1.0
sudo python setup.py install

## A.5 ADC Pi

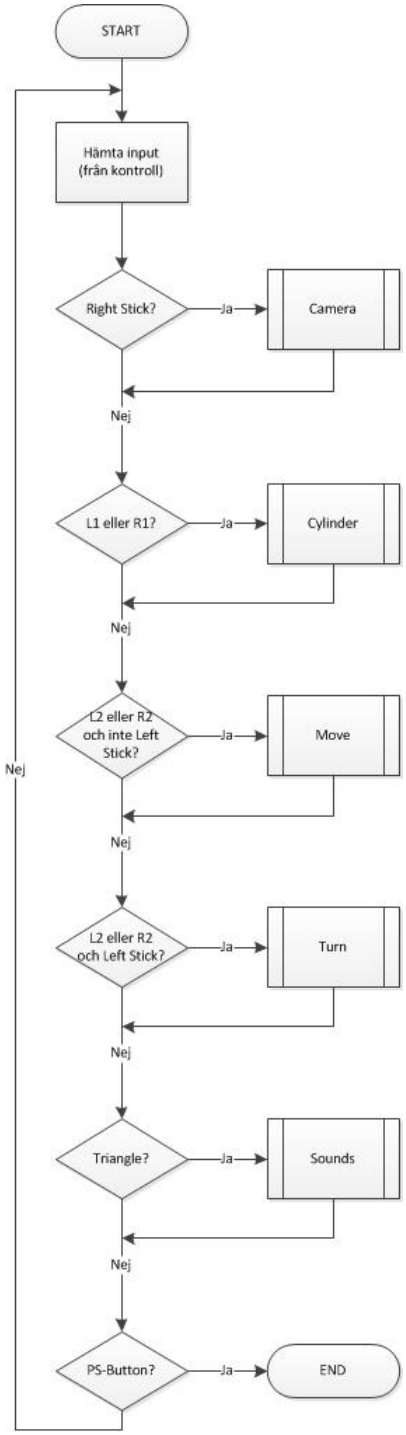
Python 3 bibliotek för ADC Pi (från AB Electronics)
sudo git clone https://github.com/abelectronicsuk/ABElectronics_Python3_Libraries.git

## A.6 Kretsschema




Figur 13: Kretsschema för robotprototypen (Författarens bild)

## A.7 Flödesschema




Figur 14: Flödesschema över main funktionen (Författarens bild)

## A.8 Kod

```
#
#       ************Gudrun************
#       Thesis Project at CTH, Chalmers Tekniska Hogskola
#
#       Python code for the Gudrun (Erwin) project'
#   at Broccoli Engineering AB, Gothenburg, Sweden.
#       Gudrun (Erwin) is a robot that moves through though terrain
#
#       By Simon Modigh and Isabelle Nilsson
#
#       Created: 06/04/15
#       Last change: 31.05.2015 18:16
#

#Imports
import threading, time, os, pygame, sys, picamera, datetime
from threading import Thread
from Adafruit_PWM_Servo_Driver import PWM
from pygame.locals import *
from ABE_ADCPi import ADCPi
from ABE_helpers import ABEHelpers


#Controller init
pygame.joystick.init()  #init
j = pygame.joystick.Joystick(0) #shortcut
j.init()        #init
j.get_init()    #init
pygame.init()   #init
events = pygame.event.get()     #shortcut
for event in events:
        UpdateMotors = 0

#Button number layout shortcuts
Select = 0
LSDown = 1      #Left Stick Down
RSDown = 2      #Right Stick Down
Start = 3
Up = 4  #D-Pad Up
Right = 5       #D-Pad Eight
Down = 6        #D-Pad Down
Left = 7        #D-Pad Left
L2 = 8
R2 = 9
L1 = 10
R1      = 11
```

```
47 Triangle = 12
48 Circle = 13
49 X = 14
50 Square = 15
51 PS = 16 #Playstation Button
52
53 #Definitions of legs and cylinders, shortcuts
54 Cyl1 = 0
55 Cyl2 = 1
56 Cyl3 = 2
57 Cyl4 = 3
58 Cyl5 = 4
59 Cyl6 = 5
60 Servo1 = 0
61 Servo2 = 1
62 Servo3 = 2
63 Servo4 = 3
64 Servo5 = 4
65 Servo6 = 5
66 CameraServo = 6
67
68 ServoPos1 = 250 #variable, value for middle
69 ServoPos2 = 250 #variable, value for middle
70 LForward = 200  #value for left side, Forward
71 LBack = 300     #value for left side, Back
72 RForward = 300  #value for right side, Forward
73 RBack = 200     #value for right side, Back
74
75 LeftTurn = 0    #Variable
76 RightTurn = 0   #Variable
77 CylIn = 460     #Cylinder up (1 ms duty cykle)
78 CylOut = 900    #Cylinder down (2 ms duty cykle)
79 Height = 550    #Position for leg
80
81 #PWM init
82 pwmServo = PWM(0x40)    #Set pwm hat #2 adress 40
83 pwmServo.setPWMFreq(40) #set the PWM frequency - servo
84 pwmCylinder = PWM(0x41) #Set pwm hat #1 adress 41
85 pwmCylinder.setPWMFreq(100)     #set the PWM frequency - cylinder
86
87 #Camera init
88 CameraAxis = 250        #Start value for the camera servo is
      attached to
89
90 #ADC init
91 i2c_helper = ABEHelpers()
92 bus = i2c_helper.get_smbus()
93 adc1 = ADCPi(bus, 0x68, 0x69, 12)       #adress for adc-pi 1
94 adc2 = ADCPi(bus, 0x6A, 0x6B, 12)       #adress for adc-pi 2
```

```
95
96  #
97  #  name: main
98  #  @param
99  #  @return
100 #  note: The main function, in endless loop
101 #
102 def main():
103         StartPosition() #Put legs in a start position
104         while True:
105                 for event in pygame.event.get():        #get
                        input from controller
106                     #Right stick
107                     if (j.get_axis(2) != 0):
108                             Camera()
109
110                     #L1 button or R1 button
111                     if j.get_button(L1) or j.get_button(R1):
112                             Cylinder()
113
114                     #L2 or R2 buttons AND NOT left stick
115                     if j.get_axis(0) == 0 and (j.get_button(
                            R2) or j.get_button(L2)):
116                             Move()
117
118                     #L2 or R2 buttons AND left stick
119                     if j.get_axis(0) != 0 and (j.get_button(
                            R2) or j.get_button(L2)):
120                             Turn()
121
122                     #Triangle button
123                     if j.get_button(Triangle):
124                             Sounds()
125
126                     #Exit program if PS button is pressed
127                     if j.get_button(PS):
128                             pygame.quit()
129                             sys.exit()
130                     pygame.event.pump()     #update queue for
                            controller
131
132
133
134 #
135 #  name: StartPosition
136 #  @param
137 #  @return
138 #  note: Used to put the legs in a starting position, called upon
139 #                start of program
```

```
140 #
141 def StartPosition():
142         pwmCylinder.setPWM(Cyl1, 0, 550)
143         pwmCylinder.setPWM(Cyl3, 0, 550)
144         pwmCylinder.setPWM(Cyl5, 0, 550)
145 
146         CylPair2In()
147 
148         ServoPair2Forward()
149 
150         pwmCylinder.setPWM(Cyl2, 0, 550)
151         pwmCylinder.setPWM(Cyl4, 0, 550)
152         pwmCylinder.setPWM(Cyl6, 0, 550)
153 
154         CylPair1In()
155 
156         ServoPair1Back()
157 
158         pwmCylinder.setPWM(Cyl1, 0, 550)
159         pwmCylinder.setPWM(Cyl3, 0, 550)
160         pwmCylinder.setPWM(Cyl5, 0, 550)
161 
162 
163 
164 
165 #
166 #  name: Move
167 #  @param
168 #  @return
169 #  note: The function to move forward in a
170 #  straight line, moves left pair then right side
171 #  can move forward or reverse depending on what
172 #  button is pressed
173 #
174 def Move():
175         #Forward, if legs isn't retracted
176         if j.get_button(R2) and not j.get_button(L2) and Height
               >460:
177 
178                 CylPair1In()
179 
180                 ServoPair1Forward()
181 
182                 ServoPair2Back()
183 
184                 CylPair1Down()
185 
186                 CylPair2In()
187 
```

```
188                 ServoPair1Back()
189
190                 ServoPair2Forward()
191
192                 CylPair2Down()
193
194                 CylPair1Down()
195
196         #Back\reverse, if legs isn't retracted
197         elif j.get_button(L2) and not j.get_button(R2) and Height
               >460:
198
199                 CylPair2In()
200
201                 ServoPair1Forward()
202                 ServoPair2Back()
203
204                 CylPair2Down()
205
206                 CylPair1In()
207
208                 ServoPair1Back()
209                 ServoPair2Forward()
210
211                 CylPair1Down()
212                 CylPair2Down()
213
214 #
215 #  name: Turn
216 #  @param
217 #  @return
218 #  note: A function to turn slightly to
219 #  the left or right as it moves forward
220 #
221 def Turn():
222         global LeftTurn
223         global RightTurn
224
225         if j.get_button(R2) and not j.get_button(L2):
226                 #Right turn if left stick is right
227                 if j.get_axis(0) > 0:
228                         RightTurn = 1
229
230                         CylPair1In()
231
232                         ServoPair1Forward()
233                         ServoPair2Back()
234
235                         CylPair1Down()
```

```
236
237                             CylPair2In()
238
239                             ServoPair1Back()
240                             ServoPair2Forward()
241
242                             CylPair2Down()
243                             CylPair1Down()
244
245                             RightTurn = 0
246                             return
247
248                     #Left turn if left stick is left
249                     elif j.get_axis(0) < 0:
250
251                             LeftTurn = 1
252
253                             CylPair1In()
254
255                             ServoPair1Forward()
256                             ServoPair2Back()
257
258                             CylPair1Down()
259
260                             CylPair2In()
261
262                             ServoPair1Back()
263                             ServoPair2Forward()
264
265                             CylPair2Down()
266                             CylPair1Down()
267
268
269                             LeftTurn = 0
270                             return
271
272
273 #
274 #  name: Cylinder
275 #  @param
276 #  @return
277 #  note: A function for controlling
278 #  the height of cylinders
279 #
280 def Cylinder():
281         global Height
282 #if R1 and not L1, and a variable check for height to not exceed
       limit
```

```
283         if j.get_button(R1) and not j.get_button(L1) and Height <
                900:
284                 Height += 3
285                 pwmCylinder.setPWM(Cyl1, 0, Height)
286                 pwmCylinder.setPWM(Cyl3, 0, Height)
287                 pwmCylinder.setPWM(Cyl5, 0, Height)
288 
289                 pwmCylinder.setPWM(Cyl2, 0, Height)
290                 pwmCylinder.setPWM(Cyl4, 0, Height)
291                 pwmCylinder.setPWM(Cyl6, 0, Height)
292 #if L1 and not R1, and a variable check for height to not exceed
       limit
293         elif j.get_button(L1) and not j.get_button(R1) and Height
                > 460:
294                 Height -= 3
295                 pwmCylinder.setPWM(Cyl1, 0, Height)
296                 pwmCylinder.setPWM(Cyl3, 0, Height)
297                 pwmCylinder.setPWM(Cyl5, 0, Height)
298 
299                 pwmCylinder.setPWM(Cyl2, 0, Height)
300                 pwmCylinder.setPWM(Cyl4, 0, Height)
301                 pwmCylinder.setPWM(Cyl6, 0, Height)
302 
303 #same as above but if already minimum height to make sure it is
       correct
304         elif j.get_button(L1) and not j.get_button(R1) and Height
                <461:
305                 CylPair1In()
306                 CylPair2In()
307 
308         pygame.event.pump()
309 
310 #
311 #  name: ServoPair1Forward
312 #  @param
313 #  @return
314 #  note: Move pair 1 servos forward
315 #
316 def ServoPair1Forward():
317         global ServoPos1
318         global RightTurn
319         global LeftTurn
320 
321         while ServoPos1 > LForward:
322                 ServoPos1 -=1
323                 Turn1=(((float(ServoPos1)/200)-1)*100)
324                 #move forward, ending point depends on variable
325                 pwmServo.setPWM(0, 0, int(ServoPos1+(LeftTurn
                       *(50-Turn1))))
```

```
326                 pwmServo.setPWM(2, 0, int(ServoPos1+(LeftTurn
                       *(50-Turn1))))
327                 pwmServo.setPWM(4, 0, int((500-ServoPos1)-(
                       RightTurn*(50-Turn1))))
328                 pygame.time.wait(5) #wait 5ms
329                 pygame.event.pump()
330
331 #
332 #  name: ServoPair1Back
333 #  @param
334 #  @return
335 #  note: Move pair 1 servos back
336 #
337 def ServoPair1Back():
338         global ServoPos1
339         global RightTurn
340         global LeftTurn
341
342         while ServoPos1 < LBack:
343                 ServoPos1 +=1
344                 Turn1=(((float(ServoPos1)/200)-1)*100)
345                 #move back, starting point depends on variable
346                 pwmServo.setPWM(Servo1, 0, int(ServoPos1+(
                       LeftTurn*(50-Turn1))))
347                 pwmServo.setPWM(Servo3, 0, int(ServoPos1+(
                       LeftTurn*(50-Turn1))))
348                 pwmServo.setPWM(Servo5, 0, int((500-ServoPos1)-(
                       RightTurn*(50-Turn1))))
349                 pygame.time.wait(5)     #wait 5ms
350                 pygame.event.pump()
351
352
353 #
354 #  name: ServoPair2Forward
355 #  @param
356 #  @return
357 #  note: Move pair 21 servos forward
358 #
359 def ServoPair2Forward():
360         global ServoPos2
361         global RightTurn
362         global LeftTurn
363
364         while ServoPos2 < RForward:
365                 ServoPos2 +=1
366                 Turn2=(((float(ServoPos2)/200)-1)*100)
367                 #move forward, ending point depends on variable
368                 pwmServo.setPWM(Servo2, 0, int((500-ServoPos2)-(
                       LeftTurn*((50-Turn2)-50))))
```

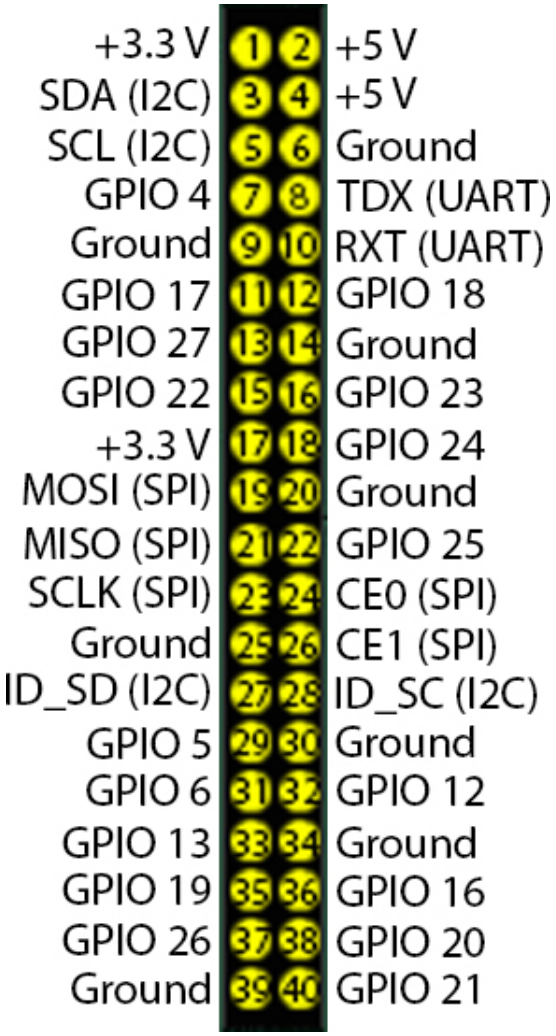
```
369                 pwmServo.setPWM(Servo4, 0, int((ServoPos2)-(
                        RightTurn*Turn2)))
370                 pwmServo.setPWM(Servo6, 0, int((ServoPos2)-(
                        RightTurn*Turn2)))
371                 pygame.time.wait(5) #wait 5ms
372                 pygame.event.pump()
373
374
375 #
376 #  name: ServoPair2back
377 #  @param
378 #  @return
379 #  note: Move pair 2 servos back
380 #
381 def ServoPair2Back():
382         global ServoPos2
383         global RightTurn
384         global LeftTurn
385
386         while ServoPos2 > RBack:
387                 ServoPos2 -=1
388                 Turn2=(((float(ServoPos2)/200)-1)*100)
389                 #move back, starting point depends on variable
390                 pwmServo.setPWM(Servo2, 0, int((500-ServoPos2)-(
                        LeftTurn*((50-Turn2)-50))))
391                 pwmServo.setPWM(Servo4, 0, int(ServoPos2+(
                        RightTurn*(((ServoPos2/200)-1)*100))))
392                 pwmServo.setPWM(Servo6, 0, int(ServoPos2+(
                        RightTurn*(((ServoPos2/200)-1)*100))))
393                 pygame.time.wait(5) #wait 5ms
394                 pygame.event.pump()
395
396
397 #
398 #  name: CylPair1Down
399 #  @param
400 #  @return
401 #  note: Move the cylinders in pair 1 down until it registers
        force
402 #
403 def CylPair1Down():
404
405         Leg1Down = False
406         Leg3Down = False
407         Leg5Down = False
408         Leg1AD = 460
409         Leg3AD = 460
410         Leg5AD = 460
411
```

```
412         #read input from ad converter, and
413         #lower leg until it reaches limit
414         while (adc1.read_voltage(1) < 4.8):
415                 Leg1AD += 2
416                 pwmCylinder.setPWM(Cyl1, 0, Leg1AD)
417                 pygame.event.pump()
418                 pygame.time.wait(25)    #wait 25ms
419         #read input from ad converter, and
420         #lower leg until it reaches limit
421         while (adc1.read_voltage(3) < 2.7):
422                 Leg3AD += 2
423                 pwmCylinder.setPWM(Cyl3, 0, Leg3AD)
424                 pygame.event.pump()
425                 pygame.time.wait(25)    #wait 25ms
426
427         #read input from ad converter, and
428         #lower leg until it reaches limit
429         while (adc1.read_voltage(5) < 3):
430                 Leg5AD += 2
431                 pwmCylinder.setPWM(Cyl5, 0, Leg5AD)
432                 pygame.event.pump()
433                 pygame.time.wait(25)    #wait 25ms
434
435
436 #
437 #  name: CylPair2Down
438 #  @param
439 #  @return
440 #  note: Move the cylinders in pair 2
441 #  down until it registers force
442 #
443 def CylPair2Down():
444
445         Leg2Down = False
446         Leg4Down = False
447         Leg6Down = False
448         Leg2AD = CylIn
449         Leg4AD = CylIn
450         Leg6AD = CylIn
451
452         #read input from ad converter, and
453         #lower leg until it reaches limit
454         while (adc1.read_voltage(2) < 3):
455                 Leg2AD += 2
456                 pwmCylinder.setPWM(Cyl2, 0, Leg2AD)
457                 pygame.event.pump()
458                 pygame.time.wait(25)    #wait 25ms
459         #read input from ad converter, and
460         #lower leg until it reaches limit
```

```
461         while (adc1.read_voltage(4) < 4.4):
462                 Leg4AD += 2
463                 pwmCylinder.setPWM(Cyl4, 0, Leg4AD)
464                 pygame.event.pump()
465                 pygame.time.wait(25)    #wait 25ms
466         #read input from ad converter, and
467         #lower leg until it reaches limit
468         while (adc1.read_voltage(6) < 0.5):
469                 Leg6AD += 2
470                 pwmCylinder.setPWM(Cyl6, 0, Leg6AD)
471                 pygame.event.pump()
472                 pygame.time.wait(25)    #wait 25ms
473
474 #
475 #  name: CylPair1In
476 #  @param
477 #  @return
478 #  note: Move the cylinders in pair 1 up
479 #
480 def CylPair1In():
481         #retract legs, continously sending until limit reached
482         while (adc2.read_voltage(1) < 3.15 and adc2.read_voltage
               (3) < 3.15 and adc2.read_voltage(5) < 3.15):
483                 pwmCylinder.setPWM(Cyl1, 0, CylIn)
484                 pygame.time.wait(5)     #wait 5ms
485                 pwmCylinder.setPWM(Cyl3, 0, CylIn)
486                 pygame.time.wait(5)     #wait 5ms
487                 pwmCylinder.setPWM(Cyl5, 0, CylIn)
488
489 #
490 #  name: CylPair2In
491 #  @param
492 #  @return
493 #  note: Move the cylinders in pair 2 up
494 #
495 def CylPair2In():
496         #retract legs, continously sending until limit reached
497         while (adc2.read_voltage(2) < 3.15 and adc2.read_voltage
               (4) < 3.15 and adc2.read_voltage(6) < 3.15 ):
498                 pwmCylinder.setPWM(Cyl2, 0, CylIn)
499                 pygame.time.wait(5)     #wait 5ms
500                 pwmCylinder.setPWM(Cyl4, 0, CylIn)
501                 pygame.time.wait(5)     #wait 5ms
502                 pwmCylinder.setPWM(Cyl6, 0, CylIn)
503
504 #
505 #  name: Camera
506 #  @param
507 #  @return
```

```
508 #  note: Controls the movement of the cameras servo
509 #
510 def Camera():
511         global CameraAxis
512         #right stick left
513         if j.get_axis(2) > 0 and CameraAxis>=120:
514                 CameraAxis -= 2
515                 pwmServo.setPWM(CameraServo, 0, CameraAxis)
516
517         #right stick right
518         elif j.get_axis(2) < 0 and CameraAxis<=410:
519                 CameraAxis += 2
520                 pwmServo.setPWM(CameraServo, 0, CameraAxis)
521
522         pygame.event.pump()
523
524
525 #
526 #  name: Sounds
527 #  @param
528 #  @return
529 #  note: Play sounds
530 #
531 def Sounds():
532         #if no sound is playing and triangle button is pressed
533         if not pygame.mixer.music.get_busy() and j.get_button(
              Triangle):
534                 #load music
535                 pygame.mixer.music.load(os.path.join('Sounds', '
                    AirHorn.mp3'))
536
537                 pygame.mixer.music.play()       #play music
538
539
540
541
542 main()  #start the main loop
```

## A.9 GPIO Layout




Figur 15: GPIO Layout (Författarens bild)

Layouten för Raspberry Pins GPIO, PWM hattarna har identisk layout. ADC Pi har bara 26 pins men är identisk på pins 1-26.