# DESIGNSPECIFIKATION

Johan Olin

Version 0.1

Status

| | | |
|---|---|---|
| Granskad | | |
| Godkänd | | |

# PROJEKTIDENTITET

Grupp 15, HT1-2015, Mr.Robot
Linköpings tekniska högskola, ISY

| Namn | Ansvar | Telefon | E-post |
|---|---|---|---|
| Johan Olin | Projektledare (PL) | 072-3055750 | johol009@student.liu.se |
| Mattias Ulmstedt | | 070-1454123 | matul773@student.liu.se |
| Per Olin | | 072-3055440 | perol834@student.liu.se |
| Hans Tchou | | 070-0535246 | hantc350@student.liu.se |
| Tor Utterborn | Dokumentansvarig (DA) | 076-8821415 | torut235@student.liu.se |
| Joacim Stålberg | | 073-9400950 | joast229@student.liu.se |

**E-postlista för hela gruppen**: johol009@student.liu.se

**Kund:** Tomas Svensson
**Kundtelefon** +46 (0)13 28 1368, tomass@isy.liu.se

**Kursansvarig**: Tomas Svensson, 3B:528, +46 (0)13 28 1368, tomass@isy.liu.se
**Handledare:** Olov Andersson

# Innehåll

# Dokumenthistorik

| Version | Datum | Utförda förändringar | Utförda av | Granskad |
|---|---|---|---|---|
| 0.1 | 2015-10-14 | Designspecifikation – Första utkast | Samtliga gruppmedlemmar | Samtliga gruppmedlemmar |

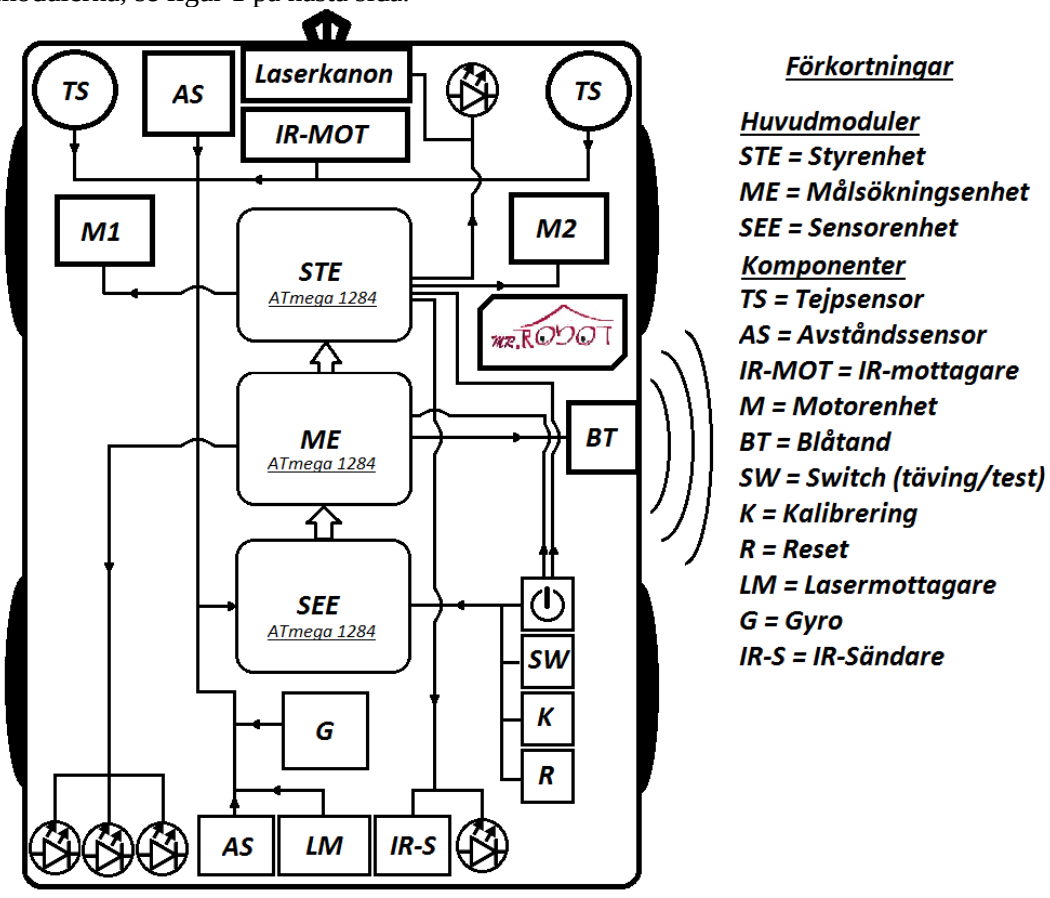
# 1 Sammanfattning

Detta dokument beskriver designen för en autonom kamprobot som del av examinationsmoment i projektkursen KMM(TSEA29 år 2015) på Linköpings universitet.
Roboten är uppbyggd av tre moduler. De tre modulerna är:

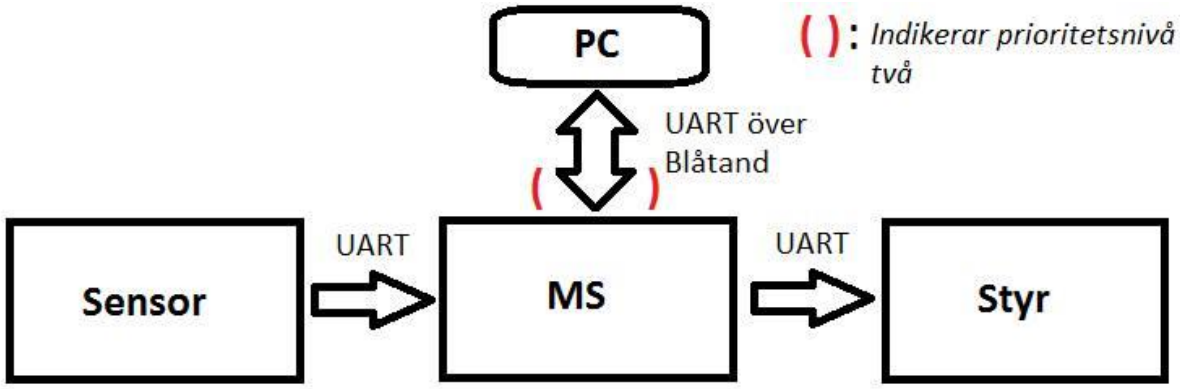
- Målsökningsenheten
- Styrenheten
- Sensorenheten

Sensorenheten tar in mätvärden med dess ingående sensorer som skickas vidare till målsökningsenheten för bearbetning av data. Målsökningsenheten tar beslut beroende på indata och skickar vidare instruktioner till styrenheten samt data till persondatorn. För att överföra data mellan enheterna så tänker vi oss främst använda överföringsmetoden UART. Styrenheten utför instruktionen. Persondatorn visar önskad mätdata på skärmen.
De ingående sensorerna och deras placering är enligt följande:

Två tejpsensorer placerade i robotens främre hörn, två ultraljudssensorer placerade fram och bak, en laserdetektor där bak, IR-sensor fram och en gyromodul placerad i robotens mitt med hänseende på y-axeln(vertikal axeln sedd uppifrån i figur 0).
De ingående ställdonen och position är enligt följande: En IR-sändare bak, lasersändare fram, motorer som är inbyggda i chassit för att styra hjulen och till sist knappar samt LEDs vars position visas i figur 0 och vars funktioner förklaras senare i dokumentet. För en mer övergripande bild utav de olika modulerna, se figur 1 på nästa sida.




*Figur 0. Bilden visar systemet i sin helhet.*




*Figur 1. Simpel bild utav systemet*

## 1.1 Gränssnittet mellan enheter

Mellan enheterna kopplar vi direkt vägar för kommunikation med UART. Vi använder Blåtand för att kommunicera mellan robot och laptop och det protokollet som används där är även UART.

### 1.1.1 UART

UART(*universal asynchronous receiver/transmitter*) är ett protokoll för hur enheter kommunicerar. UART arbetar seriellt med startbit och stoppbit.
AVR-processorerna som används har portar för att skicka (TX) och ta emot (RX) data över UART. För att använda dessa så laddas vissa register med parametrar för överföringen, till exempel BAUD-rate, antal data- och stoppbitar mm. Dessa värden bör vara likadana för alla enheter som ska kommunicera med varandra.

Vi kommer använda oss av processorernas avbrottsfunktioner för att veta när en överföring är klar. När detta händer så sparar vi över datan i receive-buffern till lokala register i den mottagande enheten.

#### 1.1.1.1 Parametrar

Vi kommer att skicka data asynkront.

| | |
|---|---|
| BAUD-rate: | 4800 |
| Databitar: | 9 |
| Stoppbitar: | 1 |
| Paritet: | ingen |
| Dubbelspeed: | nej |

#### 1.1.1.2 Vad ska skickas

**Sensorenhet → Målsökningsenhet:**

Tanke: Skicka meddelande 1 med 200 Hz och meddelande 2-3 i 50 Hz eftersom att tejpsensorerna är mest kritiska. I maxhastighet (1 m/s) med mätningar i 200 Hz så mäts med intervallet 5 mm (och tejpen är 14-18mm).

Meddelande 1:
Bit 0-1: Meddelande ID (00)
Bit 2-6: Främre avståndssensorn (ca 1 dm precision)
Bit 7: Tejpsensor 1 (vänster, 1 för tejp)
Bit 8: Tejpsensor 2 (höger, 1 för tejp)

Meddelande 2:
Bit 0-1: Meddelande ID (01)
Bit 2-6: Bakre avståndssensorn (ca 1 dm precision)

Meddelande 3:
Bit 0-1: Meddelande ID (10)
Bit 2: Aktiv IR-signatur (robot framför oss)
Bit 3-5: IR-signaturen
Bit 6: Laser (1 för träff)
Bit 7-8: 2 MSB Gyro (grader rotatation)

Meddelande 4:
Bit 0-1: Meddelande ID (11)
Bit 2-8: 6 LSB Gyro (grader rotatation)

**Målsökningsenhet → Styrenhet/PC:**

Bit 0-8: OP-kod för instruktion

### 1.1.2 Bluetooth

Roboten ska kunna skicka data till en persondator(PC) med hjälp av Bluetooth. Roboten kommer använda sig av Firefly-modulen och persondatorn kommer använda sig av en Bluetooth-pinne för att skapa en Bluetooth länk mellan de. För kunna ansluta via Firefly till en dator behöver Bluetooth-modulen använda RS232-kommunikation. Då roboten använder sig av 5 volts logik behövs en MAX232 krets för att omvandla kommunikationssignalerna till +/- 12V.

### 1.1.3 Parametrar

BAUD/BPS: 4800
Databitar: 9
Paritet: N ( ingen paritets bit)
Stoppbitar: 1

### 1.1.4 Aktivering av virtuell länk

Vi följer hänvisningarna på Vanhedens hemsida om Bluetooth

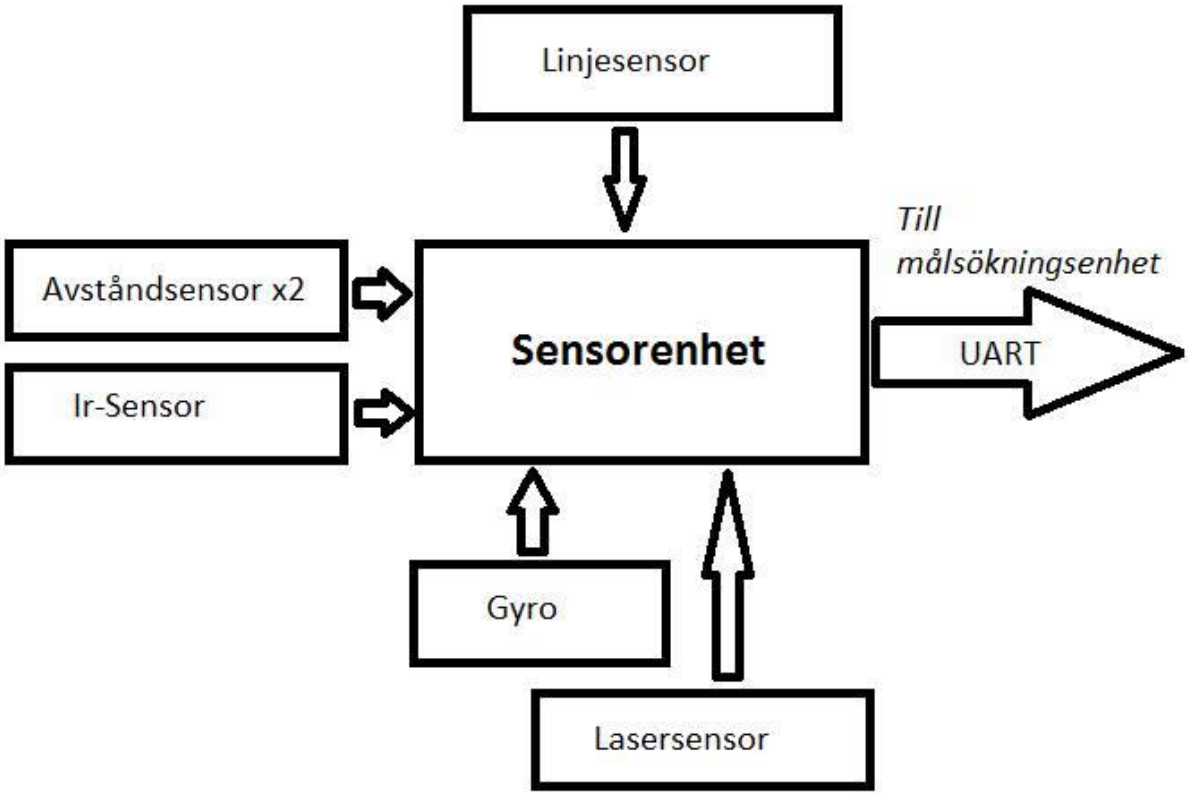
# 2 Delsystem (moduler)

I detta kapitel beskrivs de olika modulerna. Varje moduls skilda komponenter beskrivs under respektive moduls underrubrik.

## 2.1 Sensorenhet

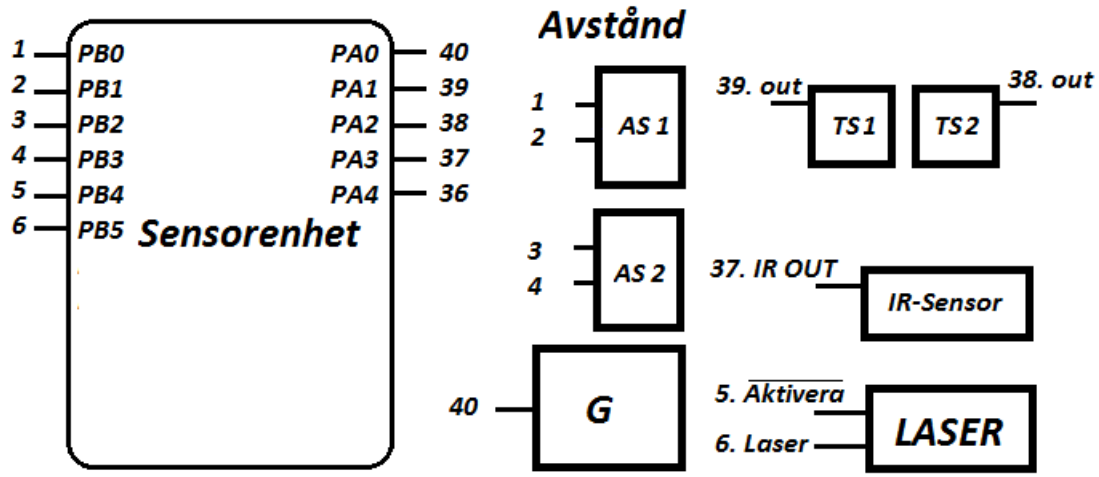
Sensormodulen är den enhet som skall sköta A-D omvandlingar från sensorvärden och skickar vidare dessa till nästa enhet. I figur 2 kan man se de ingående sensorerna. Linjesensorn används för att hålla roboten inom banans gränser.

Vi placerar avståndssensorerna fram och bak för att ta reda på om eventuella hinder befinner sig framför eller möjliga fiender bakom. IR-sensorn ska känna av motståndarnas IR signaturer (placeras fram på roboten enligt Appendix A i kravspecifikationen). Lasersensorn känner av träff från motståndare. Gyron mäter vinkelfrekvensen som kan integreras för att ta reda på hur många grader roboten roterat. Sensorerna kan tänkas skicka data i 100-1000 Hz.




*Figur 2. Blockschema över sensorenheten.*

I sensorenheten ingår ett antal sensorer och en processor. Som nämnt så sker kommunikationen till nästa enhet med UART. Processorns uppgift är i princip bara att omvandla analoga signaler till digitala signaler och skicka vidare dessa (möjligtvis omvandlas data till värden som meter, gradtal o.s.v.). Figur 3 nedan visar kopplingsschema för sensorenheten.



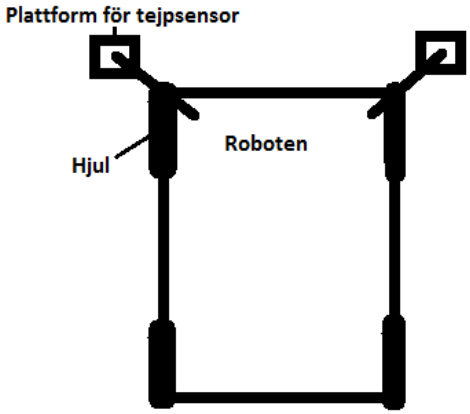

*Figur 3. Kopplingsschema för sensorenheten.*

### 2.1.1 Komponenter

- Processor (AVR ATmega 1284P)
- Tejpsensor x 2
- Avståndsensor x 2
- IR-Sensor
- Gyro
- Lasersensor

#### 2.1.1.1 Tejpsensorer

I robotens främre två hörn ska tejpsensorer (reflexsensorer) sitta som känner av om en svart eltejp befinner sig under den. Dessa sensorer ska monteras så att de går ut strax utanför roboten enligt figur 4 så att tejpen registreras innan det är för sent och roboten redan har kört över linjen.




*Figur 4: Bild på hur tejpsensorerna ska monteras*

Tejpsensorerna har 3 pinnar:

- +5: kopplas till 5V matningsspänning
- GND: kopplas till jord
- OUT: modulens analoga utsignal

Utsignalen ger en analog spänning mellan 0-5V beroende på hur mycket ljus som reflekteras på underlaget. Om underlaget reflekterar lite ljus (som i fallet för svart eltejp) så blir utsignalen hög, och om underlaget reflekterar mycket ljus så blir utsignalen låg. För användningsområdet att detektera svart tejp bryr vi oss inte om exakt hur svart tejpen är, utan vi kommer att omvandla utsignalen till en digital bit, 1 för tejp och 0 för mark.

För att sensorn ska fungera bra så måste den vara monterad så att den sitter nära marken (några millimeter). För att kalibrera sensorerna till vad som räknas som tejp och inte så ska sensorerna placeras så att båda är rakt över en tejpbit, och sedan en kalibreringsknapp tryckas. Vi sparar sedan detta analoga värde, och när sensorenheten sedan läser av sensorn för att avgöra om den befinner sig över en tejpbit eller inte så jämförs utsignalen med detta sparade värde (med viss felmarginal). Det är bättre att kolla efter tejp än att kolla efter golv, eftersom att tejpen alltid kommer att vara samma färg.

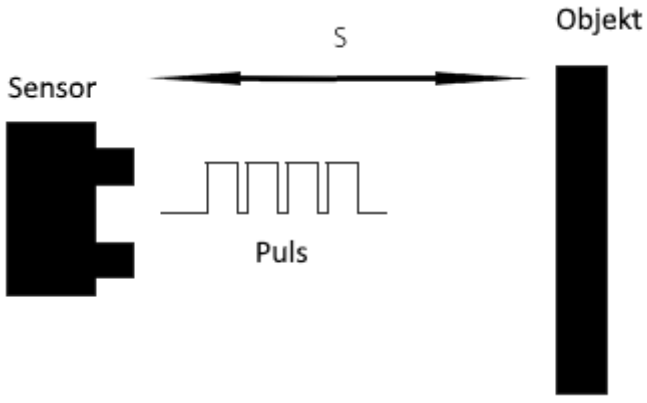
#### 2.1.1.1 Avståndsensor - SRF04 Ultrasonic Range Finder

Ultraljudssensorn ska användas för att mäta avståndet mellan objekt framför och bakom roboten. För att beräkna avstånd mellan sensor och robot använder vi formeln 1.

$$S = 170\frac{m}{s} * t$$

*Formel 1. Formeln beräknar sträckan mellan sensor och robot. S är sträckan(m), 170 är ljudets hastighet dividerat med två och t är tiden(s).*

Sensor skickat ut en puls som färdas sträckan 2S eftersom pulsen studsar tillbaka när den träffar ett objekt. När pulsen kommer tillbaka kommer tiden det tog för pulsen att färdas fram och tillbaka beräknas, se Figur 5.




*Figur 5. Bilden visar hur sensorn fungerar*

#### 2.1.1.2 IR-sensor – IRM-8601S

IR-sensorn sitter på robotens främre del där den har som uppgift att upptäcka andra kamprobotar. Sensorn detekterar IR-ljusvågor som andra robotar sänder ut från sina s.k. ”fyrar” och på så sätt identifieras motståndaren framför.

IR-sensormodulen har dessa följande 3 pinnar:

- VCC: kopplas till 5V matningsspänning
- GND: kopplas till jord
- OUT: modulens digitala utsignal

Eftersom att IR-sensorn inte kan ta emot kontinuerliga ljuspulståg för att avkodas direkt, så varierar insignalen mellan av och på med en viss period.

#### 2.1.1.3 GYRO – MLX90609 Angular Rate Sensor

Gyrot används för att beräkna om roboten roterat tillräckligt för att nå sin uttänkta rotation. Gyrot kan som maximalt mäta 300 graders rotation per sekund, där 0.5V betyder 300 grader per sekund motsols och 4.5v betyder 300 grader per sekund medsols. 2.5V betyder att det inte är någon rotation. Med hjälp av de intervallen, 0.5 – 2.5 och 2.5 – 4.5 beräknas robotens nuvarande rotationshastighet.

#### 2.1.1.4 Lasersensor

Lasersensormodulen kommer att vara monterad längst bak i mitten av roboten, och dess syfte är att detektera träff av en laser från en annan kamprobot.

Lasersensormodulen har följande pinnar:

- VCC: kopplas till 5V
- 3x GND: minst 2st bör kopplas till jord
- LASER: utgång som ger logisk 1:a vid detektering av laser
- AKTIVERA: negativ flank aktiverar laserdetektorn

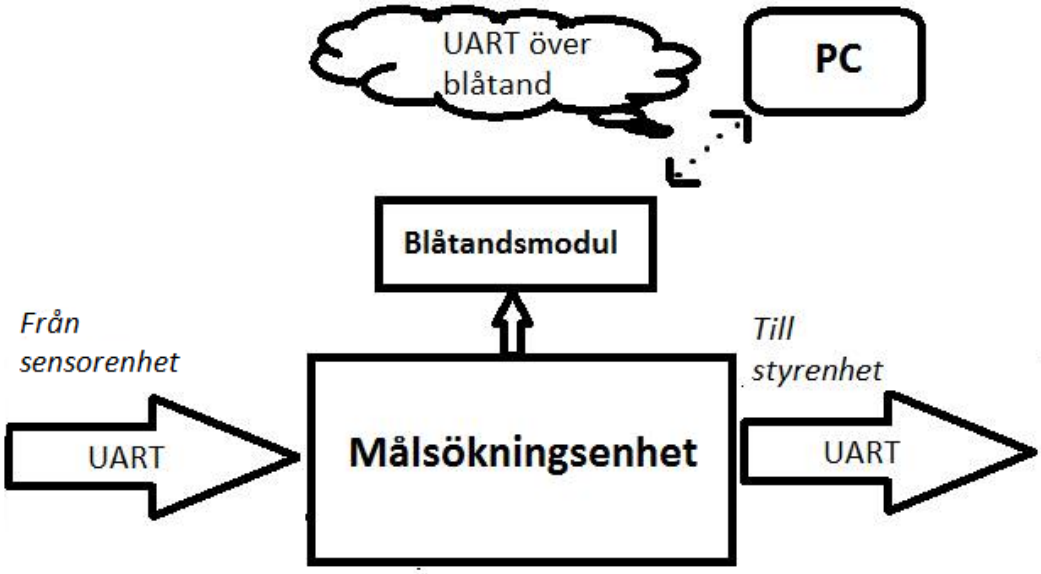
Efter spänningstillslag samt efter varje detektering av laser måste signalen *AKTIVERA* ges en fallande flank för att återaktivera detekteringen. Förutom att detektera laser kan modulen även användas som en IR-fyr, men det kommer inte att användas i det här fallet.

### 2.1.2 ADC – Analog to Digital Conversion

För att ta reda på det digitala värdet av en analog signal så jämför man två intervall. Det ena intervallet är 0V – 5V, det analoga intervallet och det andra är 0 – 255 eller 0- 1023 (beroende på om man använder 8 eller 10 bitar). 0V motsvarar 0 i det digitala intervallet och 5V motsvarar 255 eller 1023 i det digitala intervallet.

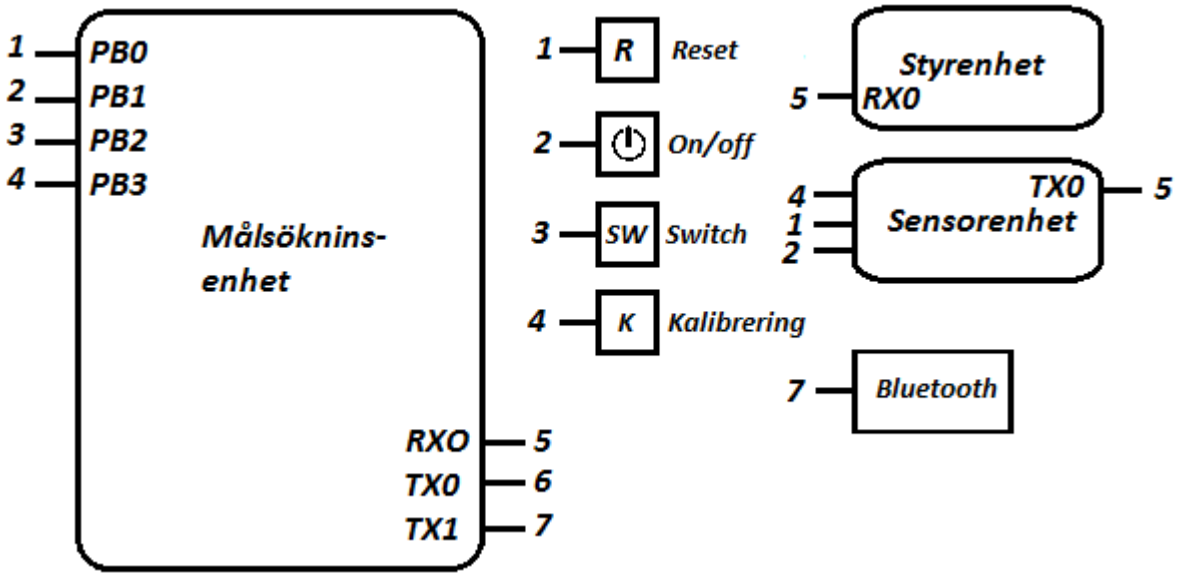
## 2.2 Målsökningsenhet

Målsökningsenheten kan ses som ”hjärnan” i roboten där huvudprogrammet kommer att existera. Algoritmen för AI samt funktioner för att skicka ut data till andra enheter existerar här. I figur 6 beskrivs i stora drag dataflödet. Data skickas med UART till både laptop och styrenhet. Kommunikationen sker direkt från och till de andra enheterna och till persondatorn sker detta via Blåtand.




*Figur 6. Blockschema över målsökningsenheten.*

Målsökningsenheten tar emot data från sensorenheten och utifrån den gör logiska val för vilka instruktioner som skickas till styrenheten. Beroende på vilket läge (test-/tävlingsläge) roboten befinner sig i ska den ha ett annorlunda beteende. Figur 7 visar kopplingsschemat för målsökningsenheten.




*Figur 7. Kopplingsschema för målsökningsenheten.*

### 2.2.1 Komponenter

- Processor (AVR ATmega 1284P)
- Blåtandsmodul
- Aktiveringsknapp
- Resetknapp
- Switch (Tävling/Test)
- Kalibreringsknapp

#### 2.2.1.1 Aktiveringsknapp

När roboten får spänning kommer den vänta på att aktiveringsknappen blir nedtryckt. Fram tills att detta händer så sitter programmet i en oändlig loop som inte gör någonting, precis innan mainloopen. Denna aktiveringsknapp kommer att vara kopplad till alla tre processorerna, vilket gör att alla kommer att starta sina program samtidigt.

```
while(ej aktiverad){
        do nothing
}
Mainloop{
        …
        …
}
```

#### 2.2.1.2 Resetknapp

När resetknappen, som är kopplad till alla tre processorerna, trycks ned kommer programmen att hoppa in i en rutin för att nollställa alla variabler (register) mm. Efter detta hoppar de upp till loopen innan mainloopen i väntan på återaktivering (genom tryck på aktiveringsknappen)

```
reset{
        nollställ allt som ska nollställas
        hoppa till loopen innan mainloopen
}
```

#### 2.2.1.3 Switch(Tävling/Test)

Roboten kommer ha ett reglage (switch) som byter mellan tävlings- och test läge.

#### 2.2.1.4 Kalibreringsknapp

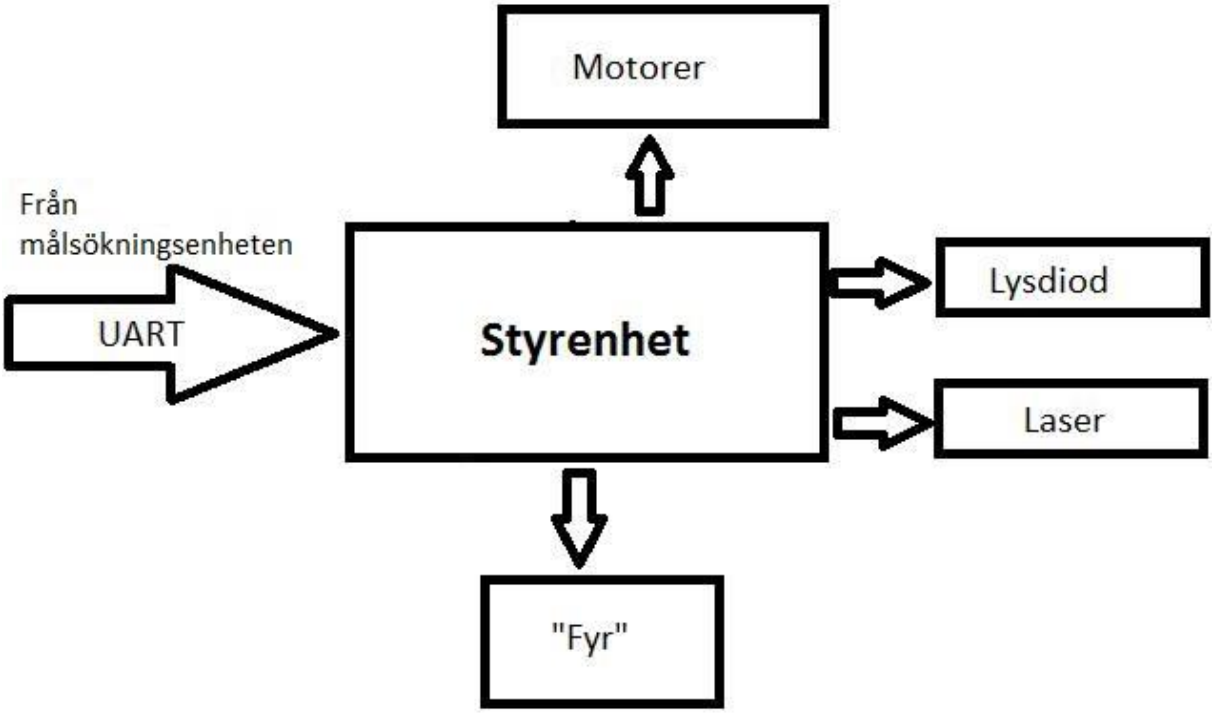
Kalibrerar tejpsensorerna.

### 2.2.2 AI

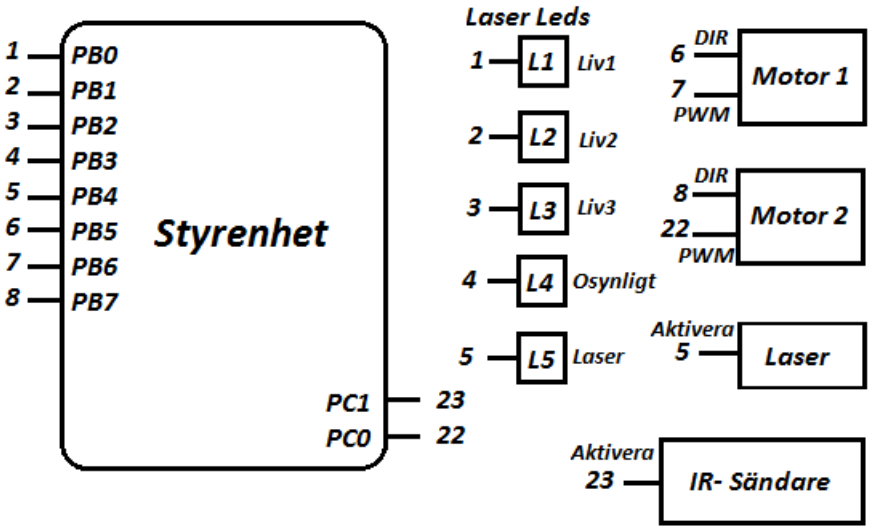
Skillnaden mellan tävlings- och testkoden är att roboten kan dö i tävlingsläget. Det medför att i tävlingsläget kommer roboten röra sig med det i åtanke medan i testläget behöver roboten inte oroa sig om detta. Denna skillnad gör att roboten kommer röra sig annorlunda i test och tävlingsläget. Se pseudokod i Appendix A för tävlings- och testkod.

## 2.3 Styrenhet

Styrenheten är den enhet som ”väcker roboten till liv”. Den styr allt rörligt och synligt på roboten, t.ex. de olika motorerna på roboten. Den får lasermodulen att skicka ut en laser samt dioderna att lysa. Den agerar efter order från målsökningsenheten och kan ses som en ”slav” till denna. Figur 8 visar blockschemat över styrenheten och figur 9 visar kopplingschemat.




*Figur 8. Blockschema över styrenheten.*




*Figur 9. Kopplingschema för styrenheten.*

### 2.3.1 Komponenter

- Processor (AVR ATmega 1284P)
- Motorer
- Laserkanon
- LEDS
- ”Fyr” - skickar IR signaturen

#### 2.3.1.1 Motorer

Roboten har 4 DC-motorer på 7.2V med upp till 291RPM, 2 på vardera sida. Vid höger respektiv vänster sväng så ska en av motorerna rotera medurs och den andra moturs för att snabba på rotationen. Motorerna ansluts till styrenheten med flatkablar och agerar utifrån order därifrån.

Motorerna styrs parvis med två signaler per sida:

- DIR: styr motorernas rotationsriktning. Kopplas direkt till microcontroller
- PWM: styr motorernas hastighet. Kopplas direkt till microcontroller

Funktioner för motorerna:

- Åka rakt fram
- Svänga höger
- Svänga vänster
- Stå still

#### 2.3.1.2 Laserkanon

Längst fram i mitten av roboten kommer en lasermodul att sitta. Denna kommer att användas för att skjuta på fyrar och fienderobotar i tävlingen.

Lasermodulen har endast 2 pinnar:

- VCC: matas med 4.5V spänning när lasern ska lysa
- GND: kopplas till jord

Eftersom att processorn bara kan ge 5V så behöver vi lägga till ett motstånd mellan VCC och matningen från processorn.

#### 2.3.1.3 LED till lasern

Denna diod kommer att tändas när lasern är aktiv. Samma signal som styr lasern styr alltså även denna LED.

#### 2.3.1.4 LED för att visa liv

Roboten ska ha tre lysdioder som representerar antalet liv roboten har kvar. Vid fullt liv lyser alla tre lysdioder och vid träff släcks en av dioderna. Lysdioderna släcks från höger till vänster.

#### 2.3.1.5 LED Osynlighet

IR sändaren ska kunna inaktiveras i 5 sekunder om roboten är osynlig.

#### 2.3.1.6 IR-sändare

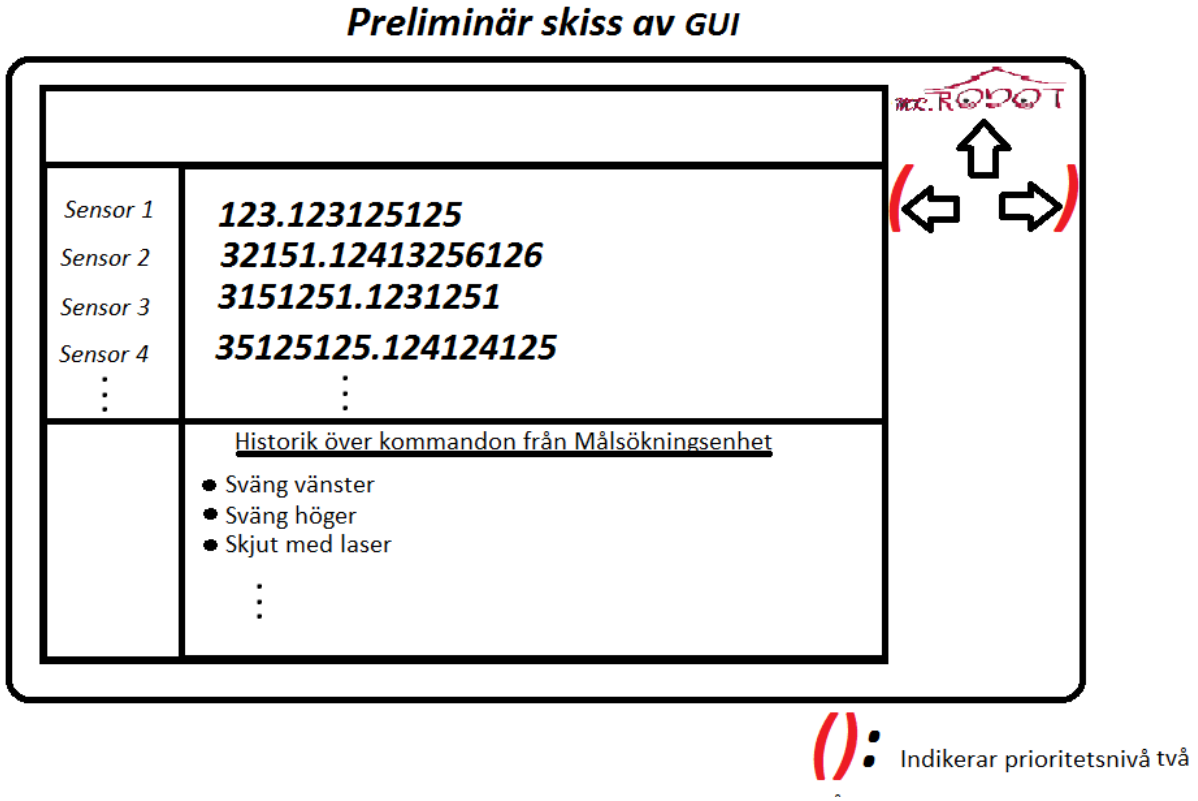
IR-sändaren (IR-fyren) sitter på bakre delen av roboten där den har som uppgift att sända ut IR-ljus som en annan kamprobot ska kunna detektera för att avgöra om vi är en fiende. IR-sändaren fungerar i princip som en vanlig LED, förutom att det är infrarött ljus som emitteras istället.
IR sändaren ska kunna inaktiveras i 5 sekunder.
IR-sändaren har 2 pinnar:

- VCC: matas med viss spänning
- GND: kopplas till jord

Matas sändaren med 5V, så kommer backspänning att uppstå.

# 3 Programvara till laptop

Programvaran tar emot data från målsökningsenheten via Bluetooth (och i mån av tid även skicka data för att t.ex. styra roboten manuellt med GUI:t). De olika sensorernas data presenteras i en lista för att fullfölja kraven i kravspecifikationen. Den nedre delen av GUI:t visar historiken över de olika kommandon roboten har utfört. Tanken är att operationshistoriken framförallt hjälper oss att ”debugga” roboten och göra arbetet med den lättare. Figur 10 visar hur vi hade tänkt oss att det grafiska användargränssnittet skulle se ut.




*Figur 10. Preliminär skiss över vårt GUI*

# 4 Implementationsstrategi

Konstruktionen sker inifrån och ut, d.v.s. vi bygger varsina moduler och komponenter och sätter sedan ihop de för att tillslut få en färdig robot.
För att se till att alla saker fungerar som vi tänkt kommer vi att prova de innan vi sätter ihop roboten. Då vi bygger roboten modul för modul är detta viktigt så man vet att alla moduler fungerar innan man gör de sista testerna med allt ihopkopplat.

Detta är något som måste ske för att vi ska kunna säga/bekräfta att en aktivitet eller milstolpe är nådd. Många av komponenterna och modulerna kan man testa enskilt utan att koppla ihop hela roboten. Detta gör att vi kan bygga roboten modul för modul. De större testen som att se till att roboten kan åka runt i en bana eller att upptäcka en fiende och skjuta mot den kräver att flera komponenter/moduler är klara.

För att testa modulerna kan man genomföra tester på varje modul, visa av modulerna kan behöva fler tester samt kräva att andra moduler fungerar. Alla sensorer, laserkanon, IR-sändare, motorer, knappar, reglage samt gyro kan testas individuellt med egna program. Dessa tester görs för att se till att modulerna fungerar men främst för att bekräfta att programmen fungerar. Dessa program kommer att fungera liknande eller vara exakt de program som vi kommer att använda vid hopkoppling och konstruktion av roboten.

Andra tester som t.ex. Bluetooth kan man också testa utan att hela systemet/konstruktionen är klar. Man kan simulera data som ska skickas via UART till Bluetooth-modulen som sedan skickar data till en persondator med vår programvara på.
För att testa styrenheten, sensorenheten och målsökningsenheten kan man göra enskilda tester. Men man kan inte vara helt säker på att allt kommer fungera innan man kopplat ihop hela roboten.
De sista testerna som sker är att se till att roboten är redo för tävling och test. Det innebär att roboten ska kunna åka runt i en bana och skjuta på andra robotar eller ”fyrar”. Om dessa tester fungerar kan vi säga att roboten är klar, redo för tävling och redovisning.

Vi kommer att skicka data från roboten till en persondator som kommer att visa alla sensor värde samt vilka operationer som styrenheten gör, åk fram, skjut laser etc. Det finns även LEDs på roboten som lyser eller är släckta beroende på olika omständigheter, visa hur mycket liv roboten har kvar eller visa att vi skjuter etc.

# Referenser

**Publicerade källor:**


- LIPS-mall för generell mall. Hämtad 2015-09-24
  http://lips.isy.liu.se/lipsmallar.html
- Svensson Thomas och Krysander Christian, 2015 Projektmodellen Lips upplaga 1:2

**Appendix.A Pseudokod**

## UART

```
Initiera UART

I sensorenheten:
          Skicka inhämtad data

I målsökning:
          Samla upp relevant data innan den används i AI delen av
koden.

          Avbrottsfunktionen:
                    Om datan börjar med 00: skicka till register x0
                    Om datan börjar med 01: skicka till register x1
                    …

I styrenheten:
          Som i målsökning men bara ett slags meddelande
```

## TEJPSENSOR

```
var tejp = 0

Omvandla till digital signal
if(digitalSignal > (kalibreradGränsSvart - felmarginal)){
          tejp = 1
}else{
          tejp = 0
}
```

## AVSTÅNDSSENSOR

```
# STARTSEKVENS
          # Skicka låg(0) till pin Pulse Trigger Input.
          # Skicka hög(1) till pin Pulse Trigger Input.
          # Sov i minst 10 µs.
          # Skicka låg(0) till pin Pulse Trigger Input.
# TIMING
          # Spara start tid.
          # När signal Echo Output blir låg(0), Spara slut tid
          # Resulterande tid = slut tid – start tid
```

# GYRO

```
Insignal är mellan 0,5 – 4,5V (konverterad gyro output)


# Variabel som lagrar procentAvMaxRotation

# Variabel som lagrar rotationsHåll

# Variabel som lagrar maxRoation

# Kolla om insignalen är större eller mindre än 2,5.

          # Om insignal är större än 2,5

                    # procentAvMaxRotation = insignal / 4.5

                    # RotationsHåll = medsols

          # Om insignal är mindre än 2,5

                    # procentAvMaxRotation = 0,5 / insignal

                    # RotationsHåll = motsols

# Utsignal = rotationsHåll + maxRotation * procentAvMaxRotation
```

### PSEUDOKOD FÖR GYRO I MÅLSÖKNINGSENHET

```
# Variabel som lagrar målRotation

# rotationshastighet = insignal från sensorehet

# rotationsTid = rotationsHastighet / målrotation

# roterar i rotationsTid sekunder åt ett visst håll ( kommer från
sensorenhet)
```

# LASERSENSOR

```
AKTIVERA = 1
AKTIVERA = 0        //aktivera sensorn
While(1){
          if(LASER == 1){
                    AKTIVERA = 1
                    Skicka data för träff till MS
                    AKTIVERA = 0        //Återaktivera sensorn
          }
}
```

## LASERKANON

```
if(skjut){
        Ge spänning till lasermodulen
        starta timer på 1 sekund
}

if(timer färdig){
        Släpp spänning till lasern
}
```

### LED FÖR LIV

```
        # antalLiv = 3

        # Om antalLiv == 3
                ## Tända alla lampor (1,2 och 3)
                # Sätt PB0, PB1 och PB2 till logisk 1
        # Om antalLiv == 2
                # Tänd lampa 1 och 2
                # Sätt PB0, PB1 till logisk 1
                # Sätt PB2 till logisk 0

        # Om antalLiv == 1
                # Tänd lampa 1
                # Sätt PB0 till logisk 1
                # Sätt PB1 och PB2 till logisk 0
```

## IR-FYR (OSYNLIGHET)

```
if(träffad){
        stäng av ir-fyren
        tänd LED
        starta timer på 5 sekunder
}

if(timer färdig){
        aktivera ir-fyren
        släck LED
}
```

### PROGRAMVARA TILL PERSONDATOR

```
        # Ta in bluetooth data
        # Omvandla data till verkliga värden och operationer
        # Uppdatera värdena som ritas ut med hjälp av ny data
```

**SWITCH (TÄVLING/TEST)**

```
          # robotMode = 0 eller 1 beroende på reglaget
          # Om robotMode == 0
                    # Kör test kod
          # Om robotMode == 1
                    # Kör tävlings kod
```

**ADC**

```
#Välj adc pinne som ska användas.

#Ställ in ADC.

#Aktivera ADC interrupts.

#Välj 8 eller 10 bitars nogranhet.

#Aktivera ADC prescaler.

#Aktivera ADC.

#Aktivera globala interupts.
```

**STYRENHET**

```
#  order = tolka och konvertera order data.
# switch(order)
          case åk frammåt
                    åkFrammåt()
          Case sväng höger
                    SvängHöger()
          Case skjut laser
                    skjutLaser()
          etc.
```

**MÅLSÖKNINGSENHET**

```
Mainloop:
          Uppdatera lokala variabler för data från sensorenheten
          Om testläge -> in i testläges rutin annars tävlingsläge
          Skicka order till styrenheten
          Skicka data till PC
```

**SENSORENHET**

```
Loop{
          filtrera data
          uppdatera data från alla sensormoduler
          Skicka data till ME
}
```

**TÄVLINGSKOD**

```
Spara in sensordata till lokala variablerna

Om tejp:
          Stanna
          Sväng

Om IR-sensorn visar fiende:
          skjut

Om bakre avståndssensorn ger utslag (och INTE främre):
          Rotera 180 grader

Om främre avståndsensorn ger utslag:
          (rotera EJ 180 grader, annars kanske de kan skjuta oss)

Om både främre och bakre ger utslag:
          Rotera 90 grader
          Fly (?)


Om lasersensorn aktiv:
          Bli osynlig
          Styr liv LEDS
          Osynlig LED?

Om vi svänger:
          Har vi svängt tillräckligt mycket?
                    Svänger = 0
          annars fortsätt svänga

Om inget intressant tas upp av sensorerna:
          Åk random / scanmode
```