Robot camp 2019

En samproduktion av

Hur man får en robot att göra som man vill!

Version 1.2

HEJSAN ZUMO ROBOT 32U4! - VAD SKA DU HETA?

Alla ZUMO robotar sätts ihop i en fabrik!

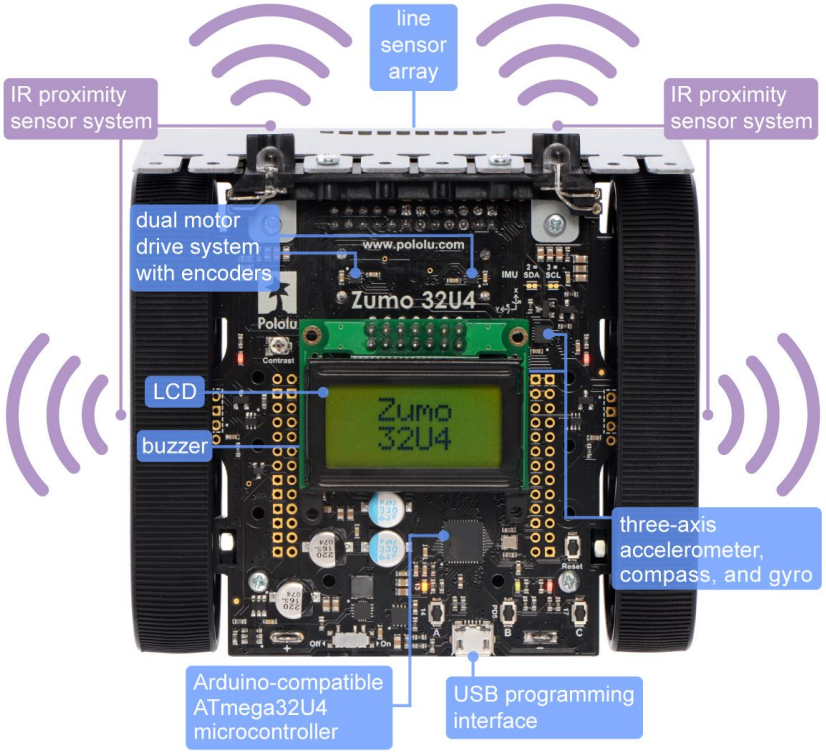
De får en liten hjärna i form av en mikroprocessor.

I början kan den ingenting!

På den här kursen ska vi lära oss hur man lär en robot!

Fundera på ett **namn** till din robot så ska vi lära den vad den heter och så att den kommer ihåg det också.

Men först ska vi lära oss programmera den!



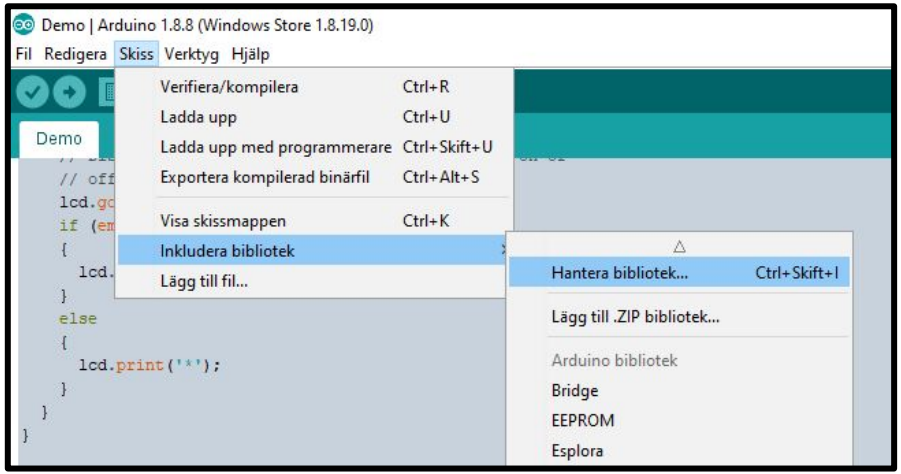
Att få arduino ide att gilla en Zumo

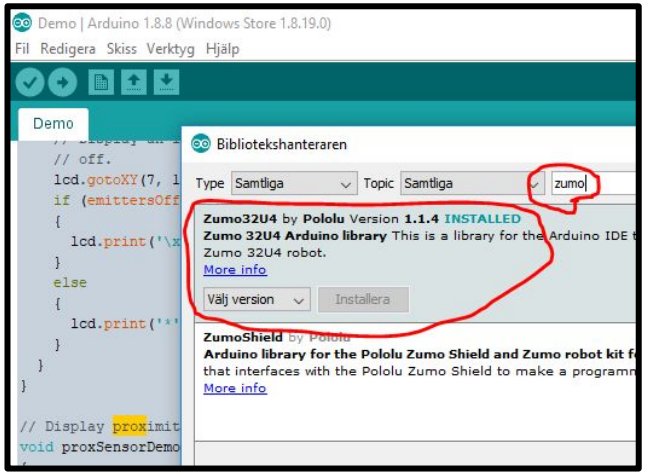
För att få Arduino IDE ska känna igen Polulu Zumo måste den anpassas litegrann:

- Installera Arduino IDE (redan gjort?)
- Installera Zumo biblioteken
- Installera Zumo robot kortet
- Välj “Pololu A-Star 32U4” under Tools menyn
- Anslut USB mellan datorn och Zumo
- Välj USB anslutning under “Tools/Port” menyn

Installera Zumo support för Arduino del I

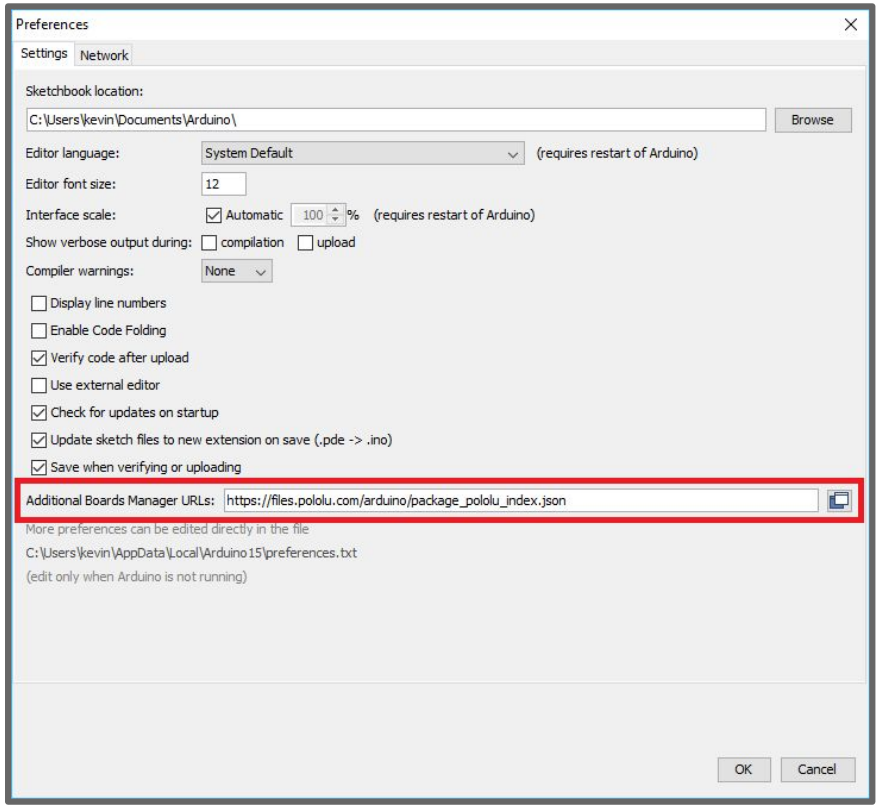
- Installera Zumo biblioteken





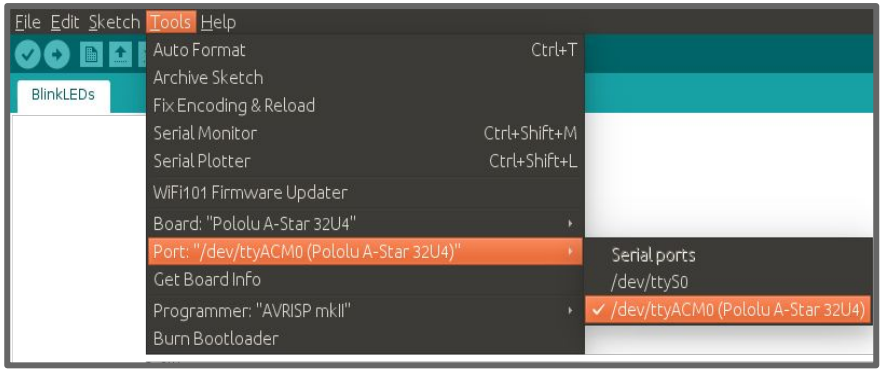
Installera Zumo support för Arduino del II

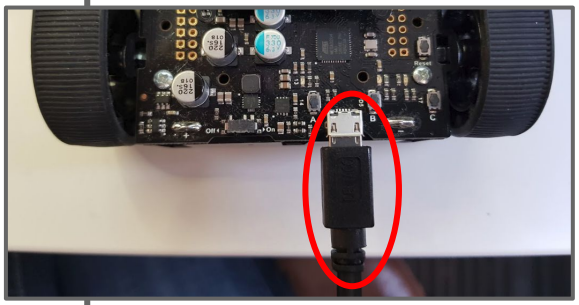
- Installera Zumo robot kortet
  - Klicka på File->Preferences
  - Fyll i “Additional Board Manager URLs:” som på bilden:



Ställ in Arduino IDE

- Välj “Pololu A-Star 32U4” under Tools menyn
- Anslut USB sladd mellan datorn och din Zumo
- Välj USB anslutning under “Tools/Port” menyn





Blinka

Din Zumo har tre stycken lysdioder.

En Röd, en Gul och en Grön.

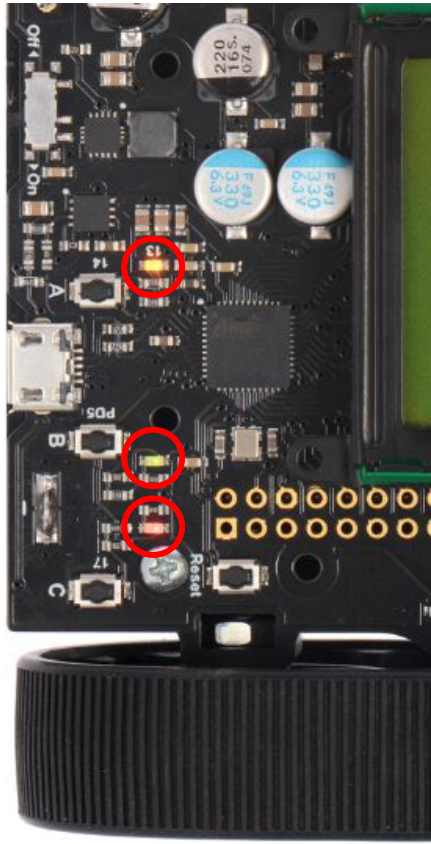
Funktionerna för tända och släcka lysdioderna heter **ledRed**() för den röda, **ledYellow**() för den gula och **ledGreen**() för den gröna.

En **1**:a tänder en lysdiod och **0**:a släcker den.

Eftersom robotens hjärna är ganska snabb måste man be den att vänta lite så den inte blinkar för snabbt med funktionen **delay**(). **1000** betyder 1 sekund.

Ni hittar exemplet i det här programmet: **Blinka**

```
/* Det här exemplet visar hur man blinkar med
lysdioderna på Zumo 32U4. */

#include <Wire.h>
#include <Zumo32U4.h>

void setup()
{

}

void loop()
{
  // Tänd lysdioderna
  ledRed(1);
  ledYellow(1);
  ledGreen(1);

  // Vänta en sekund
  delay(1000);

  // Släck lysdioderna
  ledRed(0);
  ledYellow(0);
  ledGreen(0);

  // Vänta en sekund
  delay(1000);
}
```



TALA

Din Zumo har en liten högtalare kallad buzzer

För att kunna programmera Zumos buzzer måste man skapa en buzzer, som vi kan kalla t.ex **fido**, men den kan heta vad ni vill.

När buzzern fido ska spela en **frekvens** använder man fidos funktion **playFrequency**() som vill veta frekvens (**440Hz**), hur länge (**200mS**) och hur högt (**7** av 15 steg) den ska spela.

När den börjat spela väntar vi på att den ska spela klart innan vi tar en ny ton med **delay**() funktionen.

Buzzern fido förstår faktiskt toner också om man använder dess funktion **playNote**() som istället för frekvens tar en ton funktion, t.ex **NOTE_A**() som vill veta vilken oktav (**oktav 4**) noten ska spelas i.

Prova exemplet på din Zumo!

Ändra siffrorna och se vad som händer?! Ändra mera!!

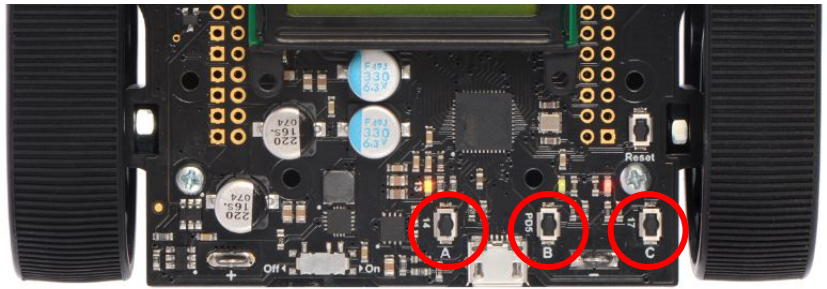
TIPS: Ändra en sak i taget och testa på roboten innan nästa ändring. Det är svårare att hitta felen om man ändrar mycket på en gång

Ni hittar exemplet i det här programmet: **Tala**

```
/*
 *  Det här exemplet visar hur man spelar
 * enkla toner på Zumo 32U4
 */

#include <Wire.h>
#include <Zumo32U4.h>

Zumo32U4Buzzer fido;

void setup()
{
}

void loop()
{
  // Spela en 440 Hz ton i 200mS på lagom volym
  fido.playFrequency(440, 200, 7);

  // Vänta en sekund (1000mS)
  delay(1000);

  // Spela tonen A i fjärde oktaven i 200mS
  fido.playNote(NOTE_A(4), 200, 7);

  // Vänta en sekund igen (1000mS)
  delay(1000);
}
```

Trycka

Din Zumo har tre (väldigt) små knappar



De har namnen **A**, **B** och **C**

För att se om en knapp är nedtryckt skapar vi först en button för varje knapp och anropar funktionen **isPressed**().Vi kallar våra buttons för **knappA**, **knappB** och **knappC** men de kan kallas vad som helst.

If-satsen testar om det är sant att knappen är nedtryckt

Ni hittar exemplet i det här programmet: **Trycka**

```
/* Det här exemplet visar hur man använder knapparna på
Zumo32U4 */

#include <Wire.h>
#include <Zumo32U4.h>

Zumo32U4ButtonA knappA;
Zumo32U4ButtonB knappB;
Zumo32U4ButtonC knappC;
Zumo32U4Buzzer buzzer;

void setup()
{
}

void loop()
{

  if (knappA.isPressed())
  {
    // Spela en 220 Hz ton i 200mS
    buzzer.playFrequency(220, 25, 15);
  }
  if (knappB.isPressed())
  {
    // Spela en 440 Hz ton i 200mS
    buzzer.playFrequency(440, 25, 15);
  }
  if (knappC.isPressed())
  {
    // Spela en 880 Hz ton i 200mS
    buzzer.playFrequency(880, 25, 15);
  }
}
```

VAD HETER DU?

Din Zumo har en liten LCD skärm kallad display

För att kunna programmera Zumos display måste man skapa en display som vi kan kalla för t.ex. **fisk**, eller vad du vill!

Zumos display är som en tavla i skolan, man måste sudda den innan man kan skriva nytt på den. När displayen fisk ska suddas använder man funktionen **clear**().

Nu kan vi skriva på displayen fisk med dess funktion **print**() som vill ha en text eller en siffra att skriva ut. För att styra vart man skriver kan man använda funktionen **gotoXY**() på displayen.
Ändra namnet **“Fisken”** till namnet på din Zumo!

EXTRA: Blinka lite med lysdioderna eller spela en trudelutt genom att lägga in kodrader från tidigare exempel.

Ni hittar exemplet i det här programmet: **Display**

```
/* Det här exemplet visar hur man skriver
på displayen */

#include <Wire.h>
#include <Zumo32U4.h>

Zumo32U4LCD fisk;

void setup()
{

}

void loop()
{
  // Sudda allt på displayen
  fisk.clear();

  // Skriv ut din robots namn
  fisk.print("Fisken");

  // Man kan skriva nästa rad också
  fisk.gotoXY(0, 1);

  // Ett nummer t.ex.
  fisk.print(1234567);

  delay(1000);
}
```

MINNS VAD DU HETER!

Din Zumo har ett litet minne kallat eeprom

För att kunna programmera Zumos minne använder du minnet **EEPROM** som är inbyggt i Zumos hjärna och kan lagra 1024 bokstäver eller små heltal med värden mellan 0 och 255.

Vi ska använda Zumos display också så vi skapar en display som heter **fisk** igen

Denna gång ska vi lagra minnet i funktionen **setup**() som bara körs en gång i början när Zumo slagits på.

Ändra **“Bettan”** till det namn du vill ha på din robot och det lagras med minnet med EEPROMs funktion **put**() som vill veta vart i minnet (position **0**) den ska lägga något och vad den ska lägga där (**“Bettan”**).

För att hämta namnet från Zumos minne måste man ha någonstans att lägga det så vi skapar en buffer som heter **“ZumosNamn”**

Sen hämtar vi namnet från Zumos minne med EEPROMs funktion **get()** som också vill veta vart i minnet (position **0**) namnet finns och vart det ska läggas (**ZumosNamn**)

Sist skriver vi ut ZumosNamn på displayen!

TESTA: Nu vet den lilla roboten vad den heter! Prova att ta bort raden med funktionen put() så får du se att den kommer ihåg namnet iallafall.

Ni hittar exemplet i det här programmet: **Minnas**

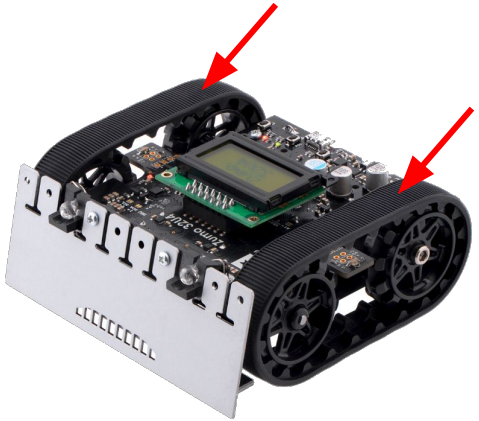

```
/*  Det här exemplet visar hur man lagrar data
 *  i minnet och sedan plockar fram det och
 *  skriver det på displayen. */
#include <EEPROM.h>

#include <Wire.h>
#include <Zumo32U4.h>

Zumo32U4LCD fisk;


// Körs en gång vid start av Zumo
void setup() {
  // Lagra namnet i minnet
  EEPROM.put(0, "Bettan");
}

// Körs gång på gång efter start av Zumo
void loop() {
  // Buffer för att hämta namnet från EEPROM
  // Måste vara längre än namnet
  // 10 lika med max 9 bokstäver i namnet.
  char ZumosNamn[10];

  // Hämta namnet från EEPROM
  EEPROM.get(0, ZumosNamn);

  // Sudda displayen fisk
  fisk.clear();

  // Skriv Zumos namn på display
  fisk.print(ZumosNamn);

  delay(1000);
}
```

Fram och tillbaka!

Din Zumo har två larvfötter drivna av motorer

För att kunna programmera Zumos att röra sig måste man skapa motorer i programmet som vi kan kalla **larver**. Det finns en motor till höger och en till vänster på Zumo och de är kopplade till larvfötterna.

I setup lägger vi nu till lite kod för att vänta på en knapptryckning så inte Sumo hoppar iväg för snabbt.



För att Zumo ska röra sig använder man funktionerna **setLeftSpeed**() och **setRightSpeed**() och de vill veta med vilken fart motorerna ska gå (**100**).

För att få Zumo att stanna anger man farten noll (**0**) till motorerna.

För att få Zumo att backa anger man en negativ fart (**-100**) till motorerna.

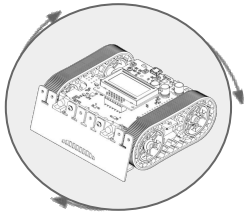
TESTA: Byt en (**-100**) mot en (**100**) någonstans i programmet. Vad händer? Varför?

Ni hittar exemplet i det här programmet: **Fram**

OBS! Glöm inte stoppa i batterierna och slå på strömmen när USB sladden är urdragen.
På knappen är Pytteliten!

```
/* Det här exemplet kör roboten fram ca 10 cm och tilllbaka */

#include <Wire.h>
#include <Zumo32U4.h>

Zumo32U4Motors larver;
Zumo32U4ButtonA buttonA;

void setup()
{
  // Vänta på att knappen A ska tryckas
  while (!buttonA.getSingleDebouncedPress());

  delay(1000); // Vänta lite till
}

void loop()
{

  // Kör framåt
  larver.setLeftSpeed(100);
  larver.setRightSpeed(100);

  delay(500);

  // Stanna
  larver.setLeftSpeed(0);
  larver.setRightSpeed(0);

  delay(500);

  // Kör bakåt
  larver.setLeftSpeed(-100);
  larver.setRightSpeed(-100);

  delay(500);

  // Stanna
  larver.setLeftSpeed(0);
  larver.setRightSpeed(0);

  delay(500);
}
```

SNURRA!



Din Zumo kan rotera och svänga!

För att programmera Zumo att rotera istället för att åka fram och tillbaka behövs bara några minustecken i programmet. Kanske kom ni på det själva?

EXTRA: Vad händer om man har olika fart på höger och vänster motor?

Ändra så att den sätter en annan fart å ena motorn, till exempel:

```
larver.setRightSpeed(-50);
larver.setLeftSpeed(150);
```

EXTRA2: Blinka med lysdioderna, pip med buzzern och skriv ut robotens namn på displayen genom att kopiera programrader från de tidigare exemplen! Lycka till!

Ni hittar exemplet i det här programmet: **Snurra**

```
/* Det här exemplet snurrar roboten fram och tillbaka */

#include <Wire.h>
#include <Zumo32U4.h>

Zumo32U4Motors larver;
Zumo32U4ButtonA buttonA;

void setup()
{
  // Vänta på att knappen A ska tryckas
  while (!buttonA.getSingleDebouncedPress());
  delay(1000);
}

void loop()
{

  // Kör framåt
  larver.setLeftSpeed(100);
  larver.setRightSpeed(-100); // <--- MINUS

  delay(500);

  // Stanna
  larver.setLeftSpeed(0);
  larver.setRightSpeed(0);

  delay(500);

  // Kör bakåt
  larver.setLeftSpeed(-100); // <--- MINUS
  larver.setRightSpeed(100);

  delay(500);

  // Stanna
  larver.setLeftSpeed(0);
  larver.setRightSpeed(0);

  delay(500);
}
```

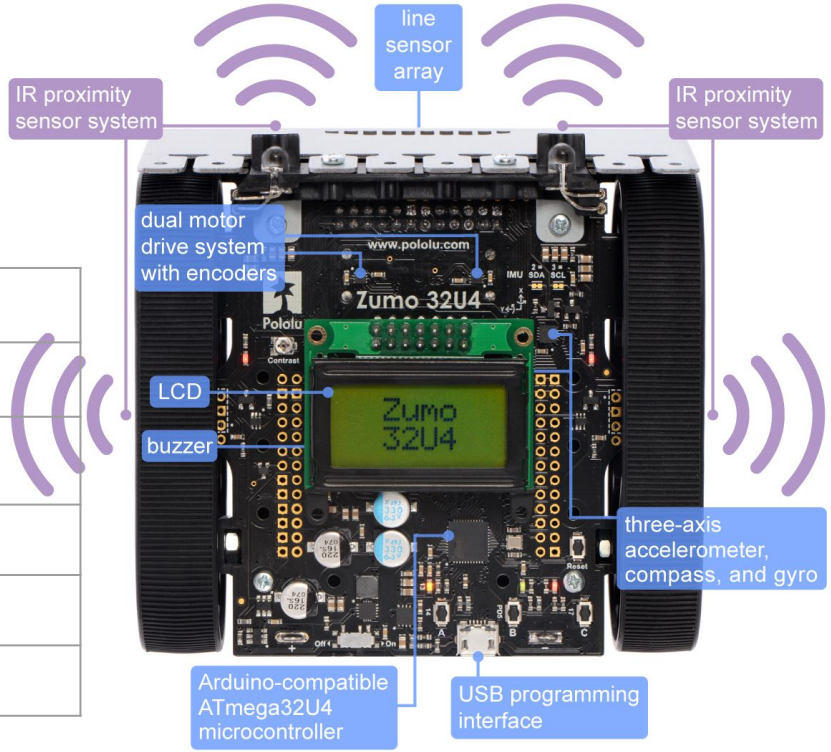
HEJSAN ZUMO ROBOT 32U4! - VAD SER DU?

Alla Zumorobotar är utrustade med en massa sensorer.

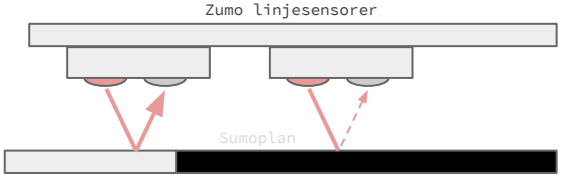
Sensorerna är robotens sinnen, precis som vi kan se, känna och lukta så kan roboten också göra det.

Nu ska vi lära roboten att se! Fundera på varför en robot måste kunna se. På bilden här bredvid kan ni se Zumos olika sensorer.

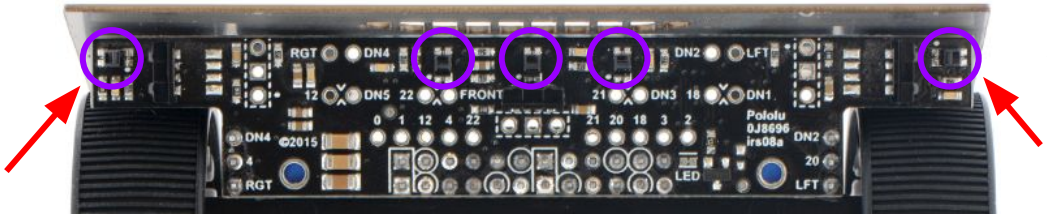
Engelskt namn	**Svenskt namn**
Line sensor array	linjesensorer eller reflexsensorer
Proximity sensor system	avståndssensorer
Accelerometer	stötsensor
Compass	kompass
Gyro	lutningssensor



Se nedåt



Din Zumo har fem stycken linjesensorer men vi ska bara använda de två som sitter längst ut till höger och vänster



För att läsa värdet på sensorerna använder man funktionen **read**() på våra **linjeSensorer** och den vill veta vart (**lineSensorVarden**) den ska lägga de värden den läser av.

QTR_EMITTERS_ON betyder att den ska lysa upp marken med IR ljus, annars ser roboten inget. Varje linjesensor har en inbyggd IR-lysdiod.

Värdena för sensorerna längst till höger (**linjeSensorVarden[4]**) och vänster (**linjeSensorVarden[0]**) skrivs ut på robotens LCD skärm och på din dators Terminal via USB kabeln.

Testa: Prova och byt till **QTR_EMITTERS_OFF** där programmet läser av med **read().** Vad händer? Varför är inte det bra?

Om ni kör fast kan ni kolla facit: **SeNed**

```
//* Det här exemplet visar hur linjesensorerna används */

#include <Wire.h>
#include <Zumo32U4.h>

Zumo32U4LCD lcd;
Zumo32U4LineSensors linjeSensorer;

void setup()
{
  linjeSensorer.initFiveSensors(); /* Det finns 5 linjesensorer fram */
}

void loop()
{
  uint16_t linjeSensorVarden[5];

  // Läs linjesensorerna.
  linjeSensorer.read(linjeSensorVarden, QTR_EMITTERS_ON);

  // Skriv ut på LCD display
  lcd.clear();
  lcd.gotoXY(0, 0);
  lcd.print("V: ");
  lcd.print(linjeSensorVarden[0]); // Linjesensorn längst till vänster
  lcd.gotoXY(0, 1);
  lcd.print("H: ");
  lcd.print(linjeSensorVarden[4]); // Linjesensorn längst till höger

  // Skriv ut seriellt på terminalen också
  Serial.print(linjeSensorVarden[0]);
  Serial.print(" ");
  Serial.print(linjeSensorVarden[4]);
  Serial.print("\n");

  delay(100); // Vänta 1/10 sekund = 100mS
}
```

SE NEDÅT OCH PIP

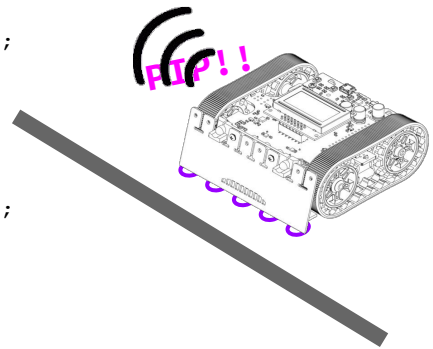
Det är svårt att läsa på LCD skärmen när roboten kör omkring så istället kan vi använda buzzern att pipa när roboten ser något.

Vi kan testa värdet från linjesensorn med en **if-sats**

```
if (linjeSensorVarden[0] < 500){ ... }
```

Prickarna mellan krullparanteserna ska ni ta bort och lägga in det som roboten ska göra om sensorvärdet är mindre än 500, t.ex. pipa lite buzzern eller blinka med lysdioderna:

```
if (linjeSensorVarden[0] < 500){
  buzzer.playFrequency(440, 50, 15);
}
```



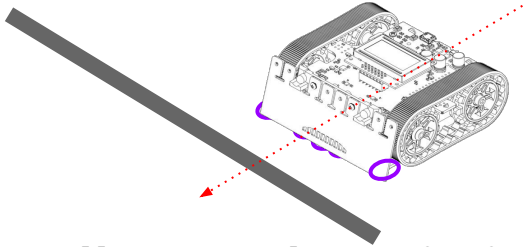
Eller det här

```
if (linjeSensorVarden[0] < 500){
  buzzer.playFrequency(440, 50, 15);
  ledRed(1);
}
else
{
  ledRed(0);
}
```

Om ni kör fast kan ni kolla facit: **SeNedPiP**


```
//* Det här exemplet visar hur linjesensorerna får roboten att pipa */

#include <Wire.h>
#include <Zumo32U4.h>

Zumo32U4LCD lcd;
Zumo32U4LineSensors linjeSensorer;
Zumo32U4Buzzer buzzer;

void setup()
{
  linjeSensorer.initFiveSensors(); /* Det finns 5 linjesensorer fram */
}

void loop()
{
  uint16_t linjeSensorVarden[5];

  // Läs linjesensorerna.
  linjeSensorer.read(linjeSensorVarden, QTR_EMITTERS_ON);

  // Pip lite
  if (linjeSensorVarden[0] < 500)
  { ... }
  if (linjeSensorVarden[4] < 500)
  { ... }

  // Skriv ut på LCD display
  lcd.clear();
  lcd.gotoXY(0, 0);
  lcd.print("V: ");
  lcd.print(linjeSensorVarden[0]); // Linjesensorn längst till vänster
  lcd.gotoXY(0, 1);
  lcd.print("H: ");
  lcd.print(linjeSensorVarden[4]); // Linjesensorn längst till höger

  // Skriv ut seriellt på terminalen också
  Serial.print(linjeSensorVarden[0]);
  Serial.print(" ");
  Serial.print(linjeSensorVarden[4]);
  Serial.print("\n");

  delay(100); // Vänta 1/10 sekund = 100mS
}
```

STANNA



Laboration:
Tejpa ett streck med den vita eller svarta eltejpen framför roboten på bordet, golvet eller Sumoplanen. Ta den färg som skiljer sig mest från underlaget.

Tag programexemplet **Fram** och gör om det så att roboten stannar vid tejpstrecket på bordet. Ni kan också ladda in **SeLinjenTemplate** som utgångspunkt.

TIPS: Kör långsamt först så att inte roboten kör förbi linjen, öka sedan så den kör så snabbt den kan utan att den missar den.

Farten kan vara mellan -400 (bakåt) och 400 (framåt)

EXTRA: Prova att köra snett mot linjen från både höger och vänster. Märker ni någon skillnad? Varför?

Om ni kör fast kan ni kolla facit: **SeLinjenFacit**

```
/*
   SeLinjenTemplate

   Modifiera det här programmet att stanna vid den tejpade linjen
   Byt ut raderna med prickar '...' mot er egen kod
*/
#include <Wire.h>
#include <Zumo32U4.h>

Zumo32U4Motors larver;
Zumo32U4LineSensors linjeSensorer;
Zumo32U4ButtonA buttonA;

void setup()
{
  linjeSensorer.initFiveSensors(); // Starta sensorerna

  while (!buttonA.getSingleDebouncedPress()); // Vänta på att knappen A
ska tryckas
  delay(1000);

  // Kör framåt
  larver.setLeftSpeed(100);
  larver.setRightSpeed(100);
}

void loop()
{
  uint16_t linjeSensorVarden[5];

  delay(10); // Läs inte sensorerna för ofta så vänta 1/100 sek

  // Läs linjesensorerna
  //... kopiera kod från tidigare exempel

  // Använd en if-sats för att testa värdet på linjesensorerna
  //... se labinstruktion för hur en if-sats ser ut
  {
    larver.setLeftSpeed(0);    // Stanna!
    larver.setRightSpeed(0);
  }
}
```

STUDSA RUNT

Laboration:
Tejpa en ring eller ruta med den vita eller svarta eltejpen på bordet, golvet eller använd Sumoplanen. Ta den tejpfärg som skiljer sig mest från underlaget.

Ta programexemplet **Fram** och gör om det så att roboten stannar vid tejpstrecket på bordet, snurrar lite och kör vidare. Eller ladda in programmet **StudsaRuntTemplate** som utgångspunkt.

EXTRA: Kör långsamt först, fart **200** är bra, öka sedan farten och testa hur roboten gör. Målet är att klara farten 400 i all olika vinklar utan att trilla över linjen.

TIPS: Ändra **delay(500)** och låt roboten snurra lite mer eller mindre. Testa hur snabbt kan den snurra!

TIPS2: Testa båda linjesenorerna för att klara alla vinklar.

TIPS3: Istället för att stanna vid linjen, ge full fart bakåt, som på en båt, en kort stund så stannar roboten snabbare och backar upp en bit på planen.

Om ni kör fast kan ni kolla facit: **StudsaRuntFacit**


```
/* Modifiera det här programmet att stanna vid den tejpade linjen
   Byt ut raderna med prickar '...' mot er egen kod
*/
#include <Wire.h>
#include <Zumo32U4.h>

Zumo32U4Motors larver;
Zumo32U4LineSensors linjeSensorer;
Zumo32U4ButtonA buttonA;

void setup()
{
  linjeSensorer.initFiveSensors(); // Starta sensorerna

  while (!buttonA.getSingleDebouncedPress()); // Vänta på att knappen A ska tryck
  delay(1000);

  // Kör framåt
  larver.setLeftSpeed(200);
  larver.setRightSpeed(200);
}

void loop()
{
  uint16_t linjeSensorVarden[5];

  // Läs linjesensorerna
  //... kopiera kod från tidigare exempel

  // Använd en if-sats för att testa värdet på linjesensorn
  //... se labinstruktion för hur en if-sats ser ut
  {

    // Bromsa
    larver.setLeftSpeed(0);    // Stanna sedan!
    larver.setRightSpeed(0);
    delay(50);                 // Vänta lite på att motorn stannar

    // Snurra lite
    larver.setLeftSpeed(200);
    larver.setRightSpeed(-200);
    delay(500);                  // Vänta tills vi snurrat klart

    // Så sätter vi fart igen
    // Kör framåt
    larver.setLeftSpeed(200);
    larver.setRightSpeed(200);
  }
}
```

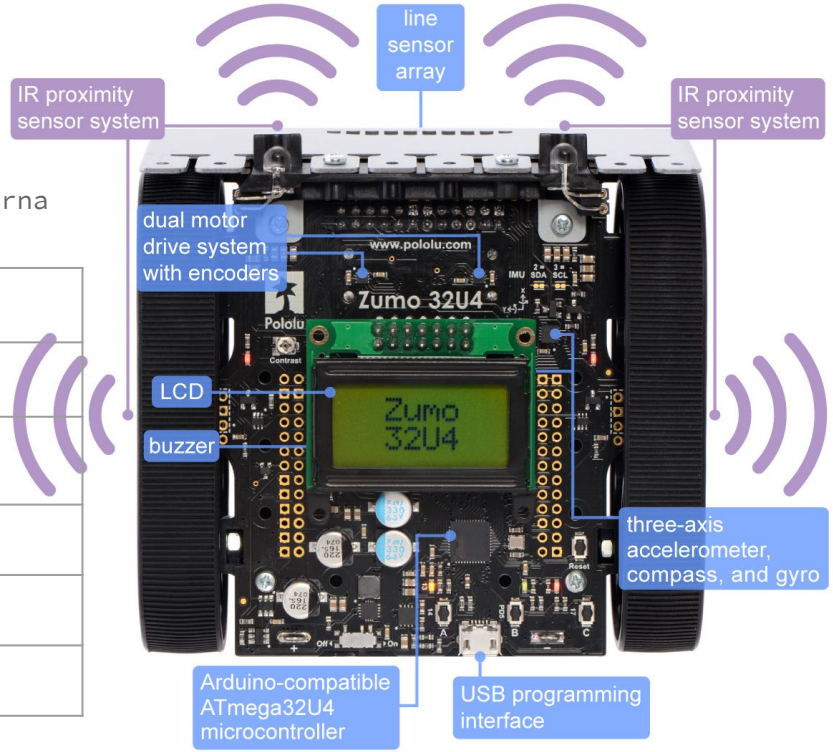
HEJSAN ZUMO ROBOT 32U4! - VAD SER DU MER?

Alla ZUMO robotar är utrustade med en massa sensorer.

Sensorerna är robotens sinnen, precis som vi kan se, känna och lukta så kan roboten också göra det.

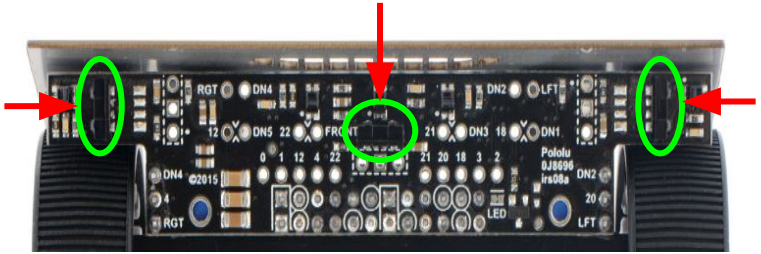
Idag ska vi lära roboten att se ännu mer! På bilden här bredvid kan ni se Zumos olika sensorer och med hjälp av avståndssensorerna kan vi hitta saker runt omkring oss.

Engelskt namn	**Svenskt namn**
Line sensor array	linjesensorer eller reflexsensorer
Proximity sensor system	avståndssensorer
Accelerometer	stötsensor
Compass	kompass
Gyro	lutningssensor

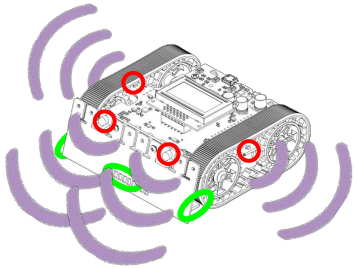


SE LJUSET

Din Zumo har **tre** stycken avståndssensorer



Till skillnad från linjesensorerna har avståndssensorerna inga inbyggda IR-lysdioder. Ljuset kommer istället från **fyra** separata IR-lysdioder:



De är av samma sort som sitter i en vanlig TV-fjärrkontroll så människor kan inte se ljuset, men roboten kan!

När IR ljuset träffar något studsar en del av ljuset tillbaka mot roboten som du kan uppfatta det med sina tre avståndsensorer. Den avståndssensor som får mest ljus tillbaka är närmast

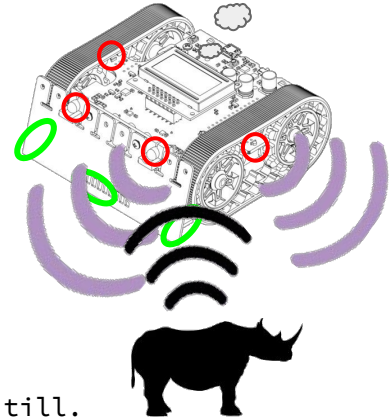
Programkoden hittar ni här: **SeLjuset**


```
//* Det här exemplet visar hur avståndssensorerna används */

#include <Wire.h>
#include <Zumo32U4.h>

Zumo32U4LCD lcd;
Zumo32U4ProximitySensors avstandsSensorer;

void setup()
{
  avstandsSensorer.initThreeSensors(); /* Det finns 3 avstånsdsensorer */
}

void loop()
{
  uint16_t avstandsSensorVarden[3];

  // Läs linjesensorerna.
  avstandsSensorer.read();
  avstandsSensorVarden[0] = avstandsSensorer.countsLeftWithLeftLeds();
  avstandsSensorVarden[1] = avstandsSensorer.countsFrontWithLeftLeds();
  avstandsSensorVarden[2] = avstandsSensorer.countsRightWithRightLeds();

  // Skriv ut på LCD display
  lcd.clear();
  lcd.print("L:");
  lcd.print(avstandsSensorVarden[0]); // Avståndssensorn längst till vänster
  lcd.print(" R:");
  lcd.print(avstandsSensorVarden[2]); // Avståndssensorn i mitten
  lcd.gotoXY(0, 1);
  lcd.print("F:");
  lcd.print(avstandsSensorVarden[1]); // Avståndssensorn längst till höger

  delay(100); // Vänta 1/10 sekund = 100mS
}
```

MER LJUS

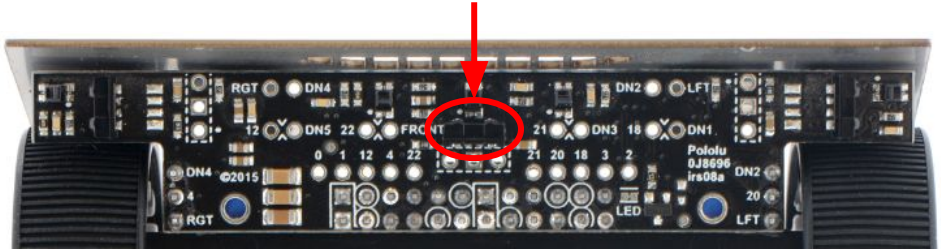
Din Zumo kan tända och släcka IR lysdioderna och mäta om det kommer tillbaka mer ljus från höger eller vänster.

Om det kommer mer ljus tillbaka om men tänder IR-lysdioderna på vänster eller höger sida så vet man att det finns något där!



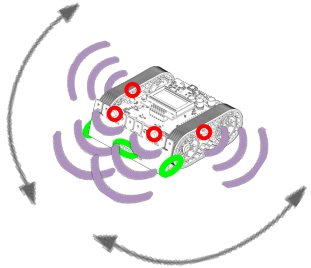
Programkoden hittar ni här: **SeMerLjus**

Fundera på vad man kan ha detta fenomen till.




```
//* Det här exemplet visar hur avståndssensoren fram kan se skillnad på
    höger och vänster*/

#include <Wire.h>
#include <Zumo32U4.h>

Zumo32U4LCD lcd;
Zumo32U4ProximitySensors avstandsSensorer;

void setup()
{
  avstandsSensorer.initThreeSensors(); /* Det finns 3 avstånsdsensorer */
}

void loop()
{
  uint16_t avstandsSensorVarden[4];

  // Läs avståndssensorerna.
  avstandsSensorer.read();

  // Avståndssensorn fram med ljus från vänster
  avstandsSensorVarden[0] = avstandsSensorer.countsFrontWithLeftLeds();

  // Avståndssensorn fram med ljus från höger
  avstandsSensorVarden[1] = avstandsSensorer.countsFrontWithRightLeds();

  // Skriv ut på LCD display
  lcd.clear();
  lcd.print("FV:");
  lcd.print(avstandsSensorVarden[0]);
  lcd.gotoXY(0, 1);
  lcd.print("FH:");
  lcd.print(avstandsSensorVarden[1]);

  delay(100); // Vänta 1/10 sekund = 100mS
}
```

SÖK



Zumo kan med hjälp av sina avståndssensorerna avgöra om det finns något till höger eller vänster snett framför.

Vi kan inkludera kod från laborationen “Snurra” och lära roboten att titta åt höger eller vänster och söka rätt på en motståndare.

Avancerat: För att programmet ska bli mer lättläst använder vi **enum** som sätter namn på tal. Till exempel FV = 0 betyder att vi kan använda FV istället för talet 0 i programmet. Man kan ange vilket namn man vill, FV kan exempelvis betyda “**F**ram **V**änster” och ange positionen för sensorvärdet för den sensorn.

Exempel på programkod för att lära roboten att använda den främre avståndssensorn hittar ni här: **SeVartfacit**

Extra: Hur kan man använda de andra avståndssensorerna som sitter på sidorna för att hjälpa roboten att vrida sig åt rätt håll? Ändra programmet så att roboten reagerar på alla avståndssensorer.

Programkod för ett exempel på lösning hittar ni här: **SeVartfacit2**


```
/* Det här exemplet visar hur avståndssensorerna används  för att bestämma
   vilket håll roboten ska vrida sig */
#include <Wire.h>
#include <Zumo32U4.h>

Zumo32U4Motors larver;
Zumo32U4ProximitySensors avstandsSensorer;
Zumo32U4ButtonA buttonA;

// Lite namn på platser i avstandsSensorVarden gör det lättare att förstå progr
enum {
  FV = 0, // Främre avståndssensor med ljus från vänster sida
  FH = 1, // Främre avståndssensor med ljus från höger sida
};

void setup()
{
  avstandsSensorer.initThreeSensors();          // Det finns 3 avståndsdsensorer
  while (!buttonA.getSingleDebouncedPress()); // Vänta på att knappen A ska try
  delay(1000);
}

void loop()
{
  uint16_t avstandsSensorVarden[4];

  avstandsSensorer.read();   // Läs avståndssensorerna.
  avstandsSensorVarden[FV] = avstandsSensorer.countsFrontWithLeftLeds();
  avstandsSensorVarden[FH] = avstandsSensorer.countsFrontWithRightLeds();

  if (avstandsSensorVarden[FV] - avstandsSensorVarden[FH] > 2)
  { // Snurra höger                    // Prova att byta 2 mot 1 eller 0, vad hä
    larver.setLeftSpeed(200); // Prova att höja farten från 200 till 400, vad h
    larver.setRightSpeed(-200);
  }
  // Om ljuset från höger syns mer än ljuset till vänster
  else if (avstandsSensorVarden[FH] - avstandsSensorVarden[FV] > 2)
  {
    // Snurra vänster
    larver.setLeftSpeed(-200);
    larver.setRightSpeed(200);
  }
  else
  {
    // Stå still
    larver.setLeftSpeed(0);
    larver.setRightSpeed(0);
  }
  delay(10); // Vänta 1/100 sekund = 10mS
}
```

KNUFFA

När Zumo hittat sin motståndare kan vi lära den att knuffa ut den utanför linjen genom att lägga till mer kod från labben “Studsa Runt”

Programexempel hittar ni här: **Knuffa**

Uppgifter: Det finns ett problem med **Knuffa** som ni snart märker, Zumo rymmer gärna utanför linjerna. Varför?

Programexempel som löser detta hittar ni här: **Knuffa2**

Nu har ni grunderna i ett komplett Sumo robot program men det saknas en hel del tricks för att vinna tävlingen och knuffa ut alla motståndare. Lägg till kod som lärt i tidigare laborationer, här är några exempel:

Knuffa3: Förbättrad knuff
Knuffa4: Använder sensorerna på sidorna
Knuffa5: Söker upp något att knuffa
Knuffa6: Använder slump för att förvirra motståndaren

Ett slumptal vet man inte på förhand vad det blir! **Random(MIN, MAX)**!

```
/* Det här exemplet visar hur avståndssensorerna används  för att bestämma
   vilket håll roboten ska vrida sig */
#include <Wire.h>
#include <Zumo32U4.h>

Zumo32U4Motors larver;
Zumo32U4ProximitySensors avstandsSensorer;
Zumo32U4ButtonA buttonA;

// Lite namn på platser i avstandsSensorVarden gör det lättare att förstå programmet
enum {
  FV = 0, // Främre avståndssensor med ljus från vänster sida
  FH = 1, // Främre avståndssensor med ljus från höger sida
};

void setup()
{
  avstandsSensorer.initThreeSensors();          // Det finns 3 avstånsdsensorer
  while (!buttonA.getSingleDebouncedPress()); // Vänta på att knappen A ska tryckas
  delay(1000);
}

void loop()
{
  uint16_t avstandsSensorVarden[4];

  avstandsSensorer.read();   // Läs avståndssensorerna.
  avstandsSensorVarden[FV] = avstandsSensorer.countsFrontWithLeftLeds();
  avstandsSensorVarden[FH] = avstandsSensorer.countsFrontWithRightLeds();

  // Om ljuset från höger syns mer än ljuset till vänster
  if (avstandsSensorVarden[FV] - avstandsSensorVarden[FH] > 2)
  {
    larver.setLeftSpeed(200); // Snurra höger
    larver.setRightSpeed(-200);
  }
  // Om ljuset från höger syns mer än ljuset till vänster
  else if (avstandsSensorVarden[FH] - avstandsSensorVarden[FV] > 2)
  {
    larver.setLeftSpeed(-200); // Snurra vänster
    larver.setRightSpeed(200);
  }
  else if (avstandsSensorVarden[FV] > 4)
  {
    larver.setLeftSpeed(200);   // Knuffa!
    larver.setRightSpeed(200);
    delay(1000);
  }
  else
  {
    // Stå still!
    larver.setLeftSpeed(0);
    larver.setRightSpeed(0);
  }
  delay(10); // Vänta 1/100 sekund = 10mS
}
```

Sumo match

När Zumo ska gå en match på sumoplanen gäller följande enkla regler:

- Två Robotar per match
- Bäst av tre ronder
- Robotarna ställs upp så här:
  - ryggen mot varann
  - vart som helst på respektive planhalva
  - ström påslagen
  - väntar på start (se kodexempel)
- När starten går väntar robotarna 3 sekunder innan de får röra sig
- Alla deltagare och åskådare flyttar sig minst 2 meter från planen för att inte störa robotarna
- Sista robot kvar på planen vinner ronden
- Domaren dömer som denne vill!

```
/* Exempel på Sumo match start */
#include <Wire.h>
#include <Zumo32U4.h>

Zumo32U4Motors larver;
Zumo32U4ProximitySensors avstandsSensorer;
Zumo32U4ButtonA buttonA;

void setup()
{
  ... // init kod

  // Vänta på att knappen A ska tryckas
  while (!buttonA.getSingleDebouncedPress())
    ;

  // Vänta 3 sekunder innan vi börjar matchen
  delay(3000);
}

void loop()
{
  .... // match kod
}
```