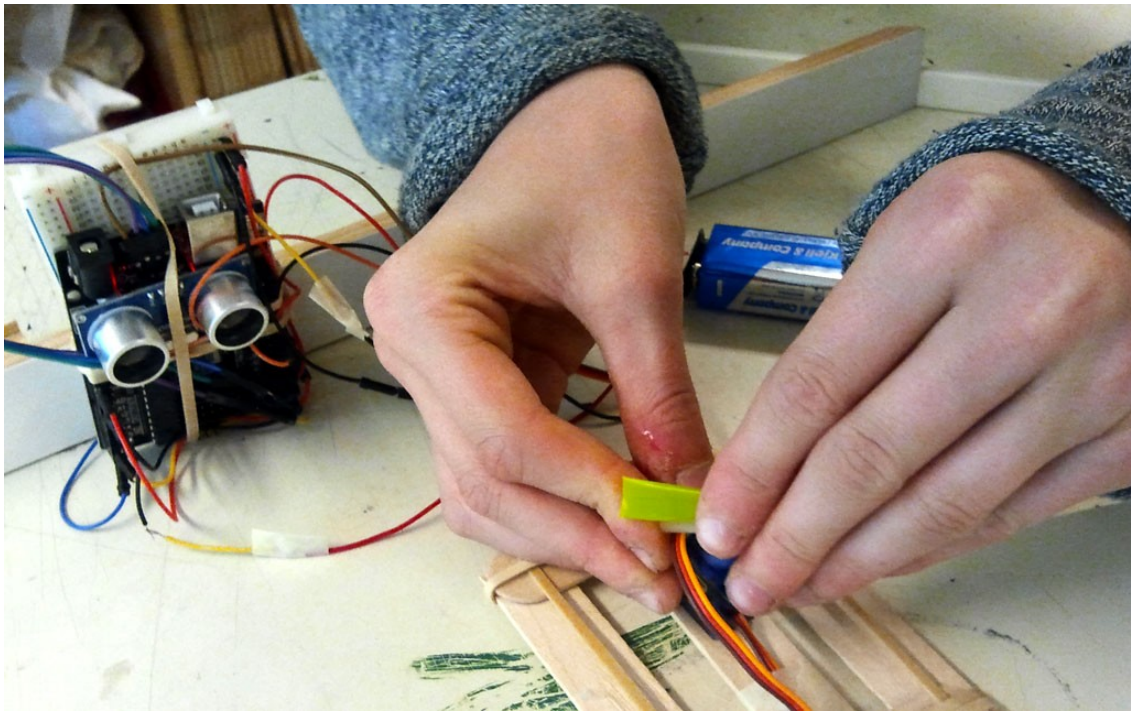
Robotfotboll med Arduino



Av Staffan Melin och Martin Blom
Bild & form-skolan, Masthugget, Göteborg
2015

Staffan Melin, staffan.melin@oscillator.se
Martin Blom, martinblomblom@hotmail.com

Teknik - Robotfotboll

Om projektet

Att bygga och programmera en enkel robot som kan slå till en boll som kommer mot den. Varje lag har 3-4 robotar som ska försöka göra poäng på motståndarlaget samtidigt som de ska förhindra ”bakåtmål”.

Tidsåtgång: 2 x 2,5h

Vi kommer att arbeta med hur **program och hårdvara jobbar tillsammans**. Det är så vår robot kan känna av och kontrollera världen runt omkring.

Att bygga något som inte är permanent är en viktig del i att ta fram en **prototyp**. Det ger möjlighet att prova sig fram.

Observera att roboten ska kunna vara återvinningsbar -- du får alltså inte limma eller löda på servot och sensorn.

Se gärna filmen: https://youtu.be/0K-a4_0Ke98.

Komponenter

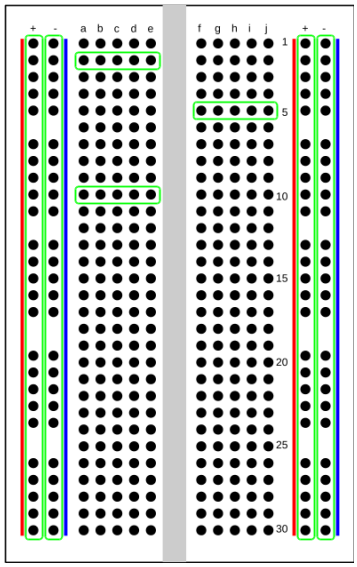
Arduino. En enkel programmerbar experimentdator som är bra på att kommunicera med omvärlden. Robotens ”hjärna”.

Sensor (avståndsmätare). Avståndsmätare använder ultraljud (ljud som har en högre frekvens än vad det mänskliga örat kan uppfatta) för att mäta avstånd. Robotens ”ögon”. Fladdermössen använder samma teknik.

Servo. Ett servo innehåller en liten motor som kan vridas åt olika håll. Robotens ”fot”.

Dator. Programmen skrivs på en dator för att sedan skickas över till Arduinon via en USB-kabel. På datorn har vi installerat en så kallad "utvecklingsmiljö", som är ett program där du kan skriva och ändra Arduino-program och med ett enkelt knapptryck skicka över dem till Arduinon via en USB-kabel.

Kopplingsbräda. Används för att koppla ihop komponenter utan att behöva löda. Hålen är sammanlänkade enligt bilden nedan. Det innebär att alla sladdar som sticks ned i hål som är inom en grön cirkel blir sammankopplade.

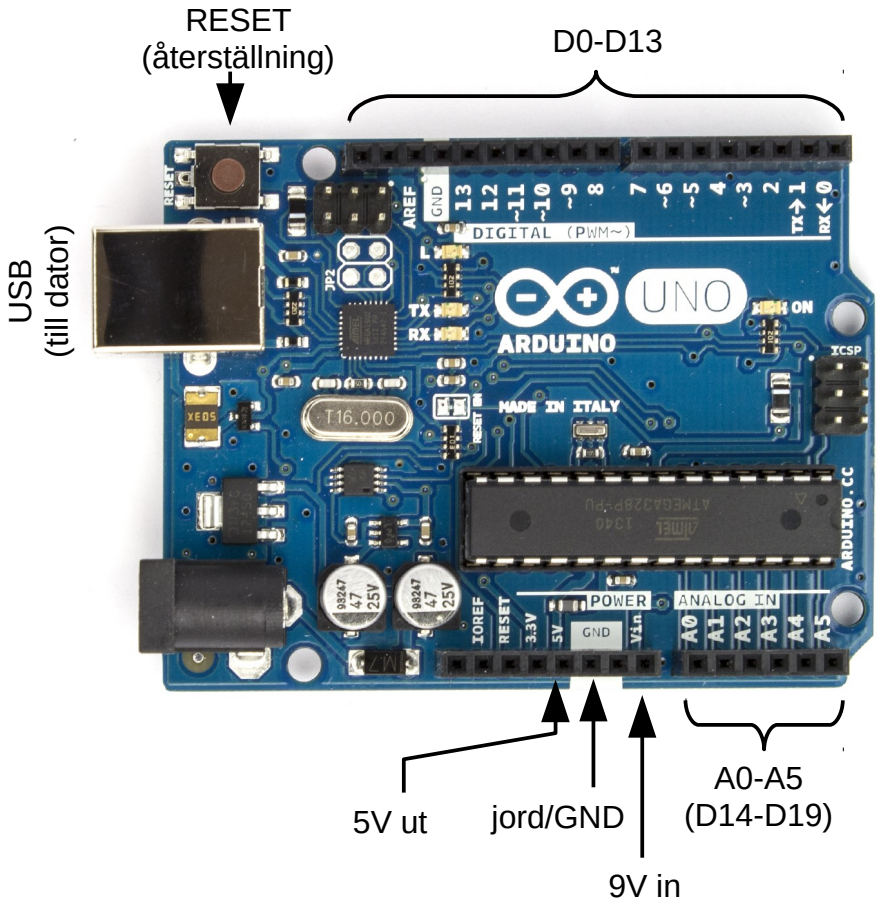


Det finns i huvudsak två olika sorters förbindelser hos varje komponent:

- de som skickar den **ström** som komponenterna behöver för att fungera (strömkrets)
- de som skickar **data** mellan komponenterna och Arduinon

Strömmen kommer från Arduinon, som i sin tur får den från datorn via USB-sladden.

För att kunna prata med omvärlden har Arduinon flera **stift eller portar**, dit du kan koppla komponenter (och sladdar):



- D0-D13: digitala stift/portar. De här kan ta emot och skicka två olika signaler: LOW (av, 0V) eller HIGH (på, +5V).
- A0-A5: analoga stift/portar. De kan ta emot värden från 0-1024 (normalt 0-5V). De kan också användas som digitala portar (och kallas då D14-D19).

Stiften kan ta emot eller skicka information - du talar om vilket i början av ditt program.

Det gäller med andra ord att dina kopplingar stämmer överens med hur du läser och skickar information i ditt program. Sladdar och komponenter samarbetar med programmet!

Beställ prylar

- Arduino Uno, inklusive USB-sladd för programmering. (Ebay: ”arduino uno R3 usb cable”, ca 50 SEK.)
- ultraljuds avståndsmätare SR03/SR04. (Ebay: ”arduino sr04”, ca 10 SEK.)
- (micro) servo motor. (Ebay: ”arduino micro servo sg90”, ca 20 SEK. Finns även i 10-pack.)

Datorer för programmering

Programmen för Arduinon skrivs på en ”vanlig” dator med hjälp av programmet **Arduino IDE**. Det kan laddas ner från Arduinons hemsida http://arduino.cc/en/Main/Software. Det finns till både Linux, Windows och Mac OSX.

Du behöver också installera biblioteket **NewPing** (detta är redan gjort innan lektionen). Du laddar ner det från playground.arduino.cc/Code/NewPing, packar upp det till en mapp, och placerar det i följande mapp: Arduino-1.0.6 > libraries (mappen ”Arduino-1.0.6” kan ha olika namn beroende på vilken version du kör). Alla bibliotek ska installeras *innan* du startar Arduino IDE.

Här kan du läsa mer om hur du installerar bibliotek: http://arduino.cc/en/pmwiki.php?n=Guide/Libraries

Del 1. Introduktion till Arduino

Vad är Arduino?

En enkel dator för experiment. Den är billig och bra på att prata med omvärlden (den har många in- och utgångar = kan kopplas ihop med annan elektronik).

- Läs mer om du vill: http://www.electrokit.com/arduino.html
- Om du vill ha ännu fler (tekniska) detaljer: http://sv.wikipedia.org/wiki/Arduino
- Arduinons hemsida: http://arduino.cc/

Vad är en robot?

"En robot är en teknisk anordning, oftast en elektromekanisk maskin som styrs av elektronisk programmering, som utför fysiska uppgifter." (Wikipedia)

Eller mer kortfattat: en maskin som styrs av en dator.

Läs mer om du vill: http://sv.wikipedia.org/wiki/Robot

Vad är ett program?

"En serie instruktioner som styr en dator " (Wikipedia)

Eller: Genom att skriva ett program talar du om för datorn vad den ska göra. Ett program skrivs ofta som en text i ett datorspråk. Det kan vara olika långt, från några rader till miljontals rader (typ JAS).

Läs mer om du vill http://sv.wikipedia.org/wiki/Datorprogram

Saker som nästan alla program har

- Funktioner. Några rader kod som kan användas flera gånger. Börjar med att de namnges.
- Variabler, konstanter. Tänk på dessa som små lådor där du kan skriva och läsa värden. Om det är en konstant börjar raden med ”const”. Då kan du inte ändra värdet senare i programmet.
- Satser/kommandon: Rader i programmet som gör saker.

Arduinon kör igång sitt program (den kan endast hålla ett program i minnet åt gången) när den får ström. När strömmen avbryts avslutas programmet (men finns kvar i Arduinons minne). Du kan starta om programmet från början genom att trycka på Arduinons RESET-knapp.

Arduino-program har två viktiga funktioner:

- **setup**. Den här delen av programmet körs en gång i början.

- **loop**. Den här delen körs gång på gång tills Arduinon inte längre får ström, eller ett nytt program laddas.

Här är ett enkelt program som får en ljusdiod att blinka. Arduinon har från början en lysdiod ansluten till port 13. Lysdioden sitter alltså redan monterad på Arduinon.

```
int pinLED = 13; // lysdioden sitter på stift 13

void setup () {
      pinMode(pinLED, OUTPUT);     // ställ in stift 13 för digital
utmatning
}

void loop () {
      digitalWrite(pinLED, HIGH);  // Slå på lysdioden
      delay(1000);                  // Vänta 1 sekund (1000 millisekunder)
      digitalWrite(pinLED, LOW);   // Slå av lysdioden
      delay(1000);                  // Vänta 1 sekund (1000 millisekunder)
}
```

Skriv in programmet på datorn och ladda upp det till Arduinon via USB-kabeln. Lysdioden ska börja blinka.

Så, vad händer på de olika raderna i programmet:

1. Vi skapar en variabel som kommer att vara konstant och innehålla värdet 13. Arduinon har en LED (lysdiod) ansluten till denna port.

2. Tom rad. Dessa struntar Arduinons i.

3-5. Setup-funktionen. Körs en gång när programmet startar. Talar om att Arduinons port med nummer 13 ska användas för att skicka information.

6. Tom rad.

7-12. Loop-funktionen. Körs om och om igen. Gör följande:

8. Sätter port 13 till PÅ. Då går en ström genom lysdioden och den tänds.

9. Väntar 1000 ms (= 1 sekund).

10. Sätter port 13 till AV. Då går inte strömmen längre genom lysdioden och den slocknar.

11. Väntar 1000 ms (= 1 sekund).

Prova att ändra värdena i delay-funktionen. Vad händer? Vad händer om du ändrar ordningen på raderna (satserna) i loop-funktionen (raderna 8-11)?

Andra viktiga saker i ett program:

- **Måsvingar**/block. Håller samman flera satser. Se http://www.arduino.cc/en/Reference/Braces.

- **for**. Upprepar satser flera gånger. Se http://arduino.cc/en/Reference/For.
- **if**. Testar ett så kallat villkor, och kör olika satser beroende på vad testet säger. Se http://arduino.cc/en/Reference/If och http://arduino.cc/en/Reference/Else.
- **case**. Påminner om if, men låter dig testa flera värden. Se http://arduino.cc/en/Reference/SwitchCase.
- **Kommentarer**. Ger möjlighet att skriva in noteringar i programmet som Arduinon sedan struntar i. Se http://arduino.cc/en/Reference/Comments.
- **Funktion**. Flera rader med satser, som ges ett namn, och kan användas flera gånger. Se http://arduino.cc/en/Reference/FunctionDeclaration.
- **random**(). En inbyggd funktion som tar fram ett slumpvärde. Se http://arduino.cc/en/Reference/Random.

Del 2: Servo

Först ska vi lära oss att styra ett **servo**. På det ska vi sedan sätta dit avståndsmätaren (**sensorn**). Ett servo är egentligen en liten motor som vi kan ställa i olika riktningar genom att ange vinkeln i grader.

Först av allt skapar vi en ström- och en jordkolumn på koppligsbrädan:

1. Dra en sladd från +5V på Arduinon till en av kopplingsbrädans +-kolumner.
2. Dra en annan sladd från GND på Arduinon till --kolumnen på kopplingsbrädan.

Nu ansluter vi servot. Servot har tre olika anslutningar:

1. Kontroll. Är ofta orange. Koppla den till port D10 på Arduinon.
2. Jord/GND. Är ofta svart/brun. Koppla den till jord-kolumnen (--kolumnen)på breadboarden
3. Ström, +5V. Är ofta röd. Koppla den till +5V-kolumnen på breadboarden

Testa nedanstående program genom att skriva in det i Arduino IDE (på datorn) och ladda upp det till Arduinon.

```
#include <Servo.h> // ett så kallat bibliotek som låter oss använda servot

int pinServo = 10; // servot är anslutet till D10

Servo myServo; // används för att kommunicera med servot

void setup()
{
  myServo.attach(pinServo);
  myServo.write(90);  // vird servot till 90 grader = rakt fram
}

void loop()
{
  myServo.write(0); // antal grader
  delay(1000); // vänta 1000 ms så servot hinner vrida sig

  myServo.write(180);
  delay(1000);
}
```

Fundera över hur programmet och kopplingarna samverkar.

Prova att ändra graderna och väntetiden (den anges i ms, millisekunder).

Spara programmet med namnet “robot_servo”. Då skapas en mapp med samma namn som programmet.

Läs mer

- http://playground.arduino.cc/Learning/SingleServoExample

Del 3. Sensor

Nu ska vi lära oss att mäta avstånd med hjälp av sensorn.

Sensorn har fyra anslutningar. På sensorn kan du se vilken sladd som är vilken anslutning. Koppla in sensorn:

1. Ström (VCC). Koppla den till +5V-kolumnen på kopplingsbrädan.
2. Styrsignal (Trig). Koppla den till D19 på Arduinon .
3. Datasignal/avståndet (Echo). Koppla den till D18 på Arduinon.
4. Jord. Koppla den till GND-kolumnen på kopplingsbrädan.

Så fungerar sensorn. Programmet skickar en signal till sensorn via Trig-porten. Då skickas en ultraljudpuls ut. Den studsar mot något föremål och kommer sen tillbaka till sensorn.

Eftersom

s = v * t, dvs sträckan = hastigheten * tiden

och ljudets hastighet i luft är något vi känner till, kan programmet räkna ut sträckan (avståndet).

Men allt det sker i ett **bibliotek** (engelska library), så vi behöver inte tänka på det. Ett bibliotek är en mängd funktioner som skrivits av någon annan, och som vi kan återanvända. Det bibliotek som vi ska använda heter **NewPing**.

Testa nedanstående program genom att skriva in det i Arduino IDE (på datorn) och ladda upp det till Arduinon.

```
#include <NewPing.h>

int pinLed = 13; // lysdiod

int pinPingTrig = 19; // sensorn ansluten till stift/port 18 och 19
int pinPingEcho = 18;
int pingMax = 500; // strunta i hinder mer än 500 cm bort

// skapa variabel (objekt) som låter oss ”prata” med sensorn
NewPing myPing(pinPingTrig, pinPingEcho, pingMax);

int avstandFramat; // vi stoppar in det uppmätta avståndet här

void setup() {

      pinMode(pinLed, OUTPUT);
}

void loop() {

      avstandFramat = myPing.ping_cm();
```

```
      if (avstandFramat < 25) // om avståndet är mindre än 25 cm
      {
            digitalWrite(pinLed, HIGH); // tänd lysdioden
      } else {
            digitalWrite(pinLed, LOW); // släck lysdioden
      }

      delay(100); // vänta 0,1 s innan vi kollar avståndet igen

}
```

Programmet fungerar så att sensorn mäter avstånd till närmaste föremål och tänder lysdioden om det är mindre än 25 cm till detta. Prova med olika avstånd.

Fundera över hur programmet och kopplingarna samverkar.

Spara programmet med namnet “robot_sensor”. Då skapas en mapp med samma namn som programmet.

Läs mer

- playground.arduino.cc/Code/NewPing

Del 4. Vi sätter ihop allt!

Om du vill, försök kombinera de två ovanstående programmen till ett eget. Det ska mäta avståndet och ”slå till” med servot om det upptäcker ett objekt.

Eller så utgår du ifrån det här programmet:

```
#include <NewPing.h>
#include <Servo.h>

int pinServo = 10;
Servo myServo;

int pinPingTrig = 19;
int pinPingEcho = 18;
int pingMax = 500;
NewPing myPing(pinPingTrig, pinPingEcho, pingMax);

int avstandFramat;

void setup()
{
  myServo.attach(pinServo);
  myServo.write(90);
}

void loop()
{

  avstandFramat = myPing.ping_cm();

  if (avstandFramat < 25) {

    myServo.write(0); // antal grader
    delay(500); // vänta 1000 ms så servot hinner vrida sig
    myServo.write(90); // antal grader
    delay(100);

  }

  delay(50);
}
```

Att fundera över

Du får prova dig fram till en lämplig konstruktion!

1. Vad ska roboten slå till med?
2. Hur ska den mäta avståndet (riktning?)
3. När ska den slå till efter att något objekt uppmätts?

- Och - ni är 4 robotar på ett lag. Ska ni anpassa robotarna till olika positioner på planen?

Spelplan

Så här byggde vi upp vår plan. Eftersom uppgiften ger stor frihet i konstruktion av robotarna, anpassa gärna planen för att maximera spelet (= chansen att robotarna träffar bollen).

Vi hade också regler om att robotarnas basyta (stödet mot marken) inte fick vara större än 5x5 cm, men vi höll inte särskilt hårt på den regerln. Det är även bra att sätta en maxlängd på armen/foten, runt 15 cm.