2014

Dustino

Enklare Robotar

2014-03-23

1 Innehållsförteckning

2 Versionshistorik

2.1 Dokument

Version	Datum	Författare	Ändringar
A	2014-03-23	Jonas Jarvoll	Första versionen

2.2 Dustino

Version	Datum	Författare	Ändringar
A	2014-03-16	Jonas Jarvoll	Första versionen

3 Förord

Grattis till att du har bestämt dig att lära dig mer om robotar! Vi tycker själva att robotar är bland det mest roliga man kan syssla med men det är svårt att förklara och sätta fingret på varför det är roligt. Kanske har det med att man för en kort stund skapar något som kommer till ”liv” eller att själva byggandet av robotar täcker in många olika områden från elektronik, mekanik till design och kreativitet. En sak är vi dock säkra på: Har man väl fått ihop sin första robot och sett den ta sina första rullande steg är det oerhört svårt att sluta bygga.

Dustino är en utmärkt plattform att använda i ditt första robotprojekt eftersom den baseras på Arduino som är välkänt för att vara speciellt inriktade mot nybörjare och det finns mycket dokumentation och guider att läsa. Med hjälp av flertal praktiska exempel i detta dokument och på vår hemsida kan du direkt komma igång med ditt skapande och på bara en kort stund känna glädjen när din robot vaknar upp. Dessutom får du nytta av din robot även efteråt då den utan att klaga ser till att hålla ditt rum rent och snyggt.

Dustino har tagits fram genom ett nära samarbete mellan de svenska robotföretagen Drones Networking och Hobbytronik tillsammans med en rad olika teknikskolor som bidragit med kunskaper och idéer. Vi hoppas att vårt jobb med Dustino inte bara ska väcka ditt intresse för robotar utan även bidra till större förståelse av robotteknik inom skolorna.

4 Introduktion

Dustino ersätter det styrkort som sitter i robotdammsugaren Centurion modell RDE125. Genom att byta ut det befintliga styrkortet får du en robotdammsugare som du själv kan programmera, styra och kontrollera. Alla funktioner som finns i robotdammsugaren (undantag laddning) stöds av Dustino styrkortet.

Dustino är Arduino kompatibelt och använder dess utvecklingsmiljö som är speciellt anpassad för nybörjare eller avancerad användare som snabbt vill komma igång med programmeringen. Det är också fullt möjligt att programmera Dustino direkt utan att gå via Arduinos utvecklingsmiljö. Detta kräver dock en extern programmerare.

- Mikrokontroller: ATmega32u4
- Kompatibel med Arduino Leonardo
- Logikspänning: 5 V
- Batteri: 14.4 V, 800 mAh (vid extern matning klarar roboten upp till 20 V)
- Flashminne: 32 KB av vilka 4 KB används av bootloadern
- SRAM: 2.5 KB
- EEPROM: 1 KB
- Frekvens: 16 MHz
- 3 st trappsensorer
- 1 st stötsensor
- Mätning av batterispänningen
- 2 st motor och hjul
- 2 st sidoborstar
- 1 st dammsugarmotor
- USB kontakt av typen B
- Reset-knapp
- ISP kontakt för direktprogrammering

5 Installation

5.1 Hårdvara

Dustino kommer i form av ett separat kort tillsammans med robotdammsugaren och du får själv montera kortet i roboten. Detta ger dig en chans att se hur roboten ser ut inuti och lära känna din robot bättre.

Att öppna robotdammsugaren och montera styrkortet är inget svårt och kräver bara en stjärnskruvmejsel.

1. Börja med att skruva bort stötfångaren, både över- och underdelen
2. Överdelen av roboten sitter fast i 5 skruvar som sitter på undersidan
3. Lirka loss alla kontakter från styrkortet
4. Ta bort styrkortet som sitter fäst med 3 skruvar
5. Montera dit Dustino på samma sätt som föregående styrkortet. Obs, spara det gamla styrkortet eftersom du använder det för att ladda roboten
6. Sätt dit kontakterna igen
7. Om du vill montera tillbaka överdelen måste du ta upp hål för reset-knappen och USB kontakten.
8. Montera till slut tillbaka stötfångaren.

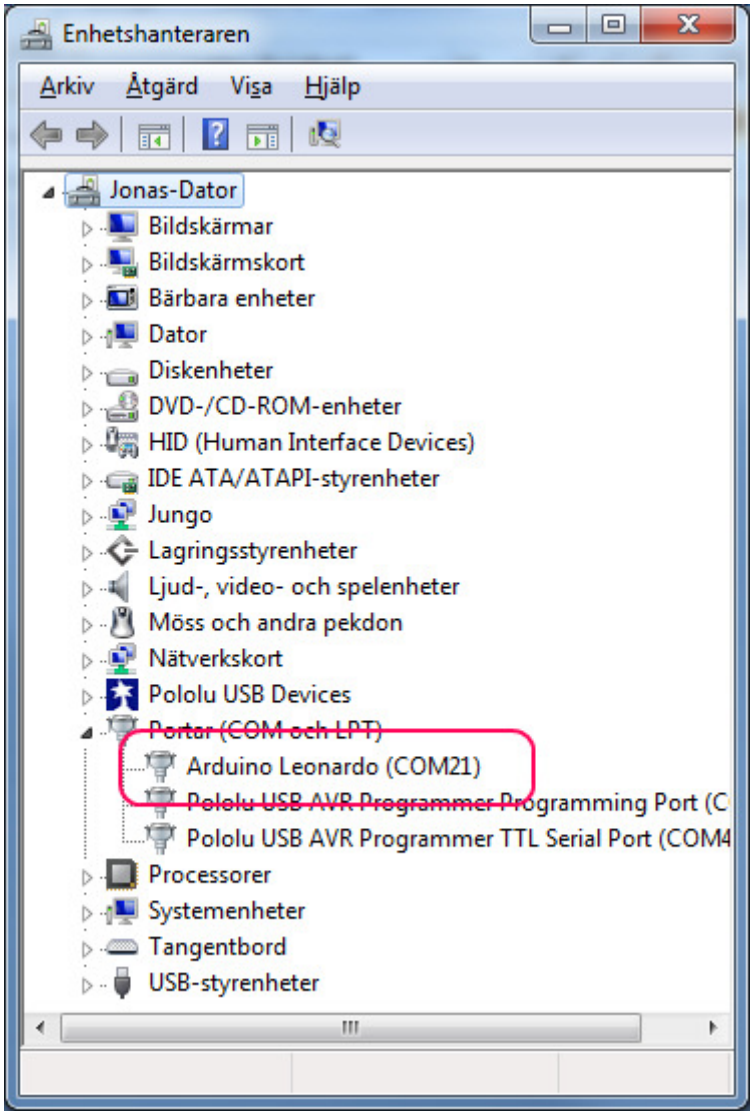
Se gärna till att roboten är laddad innan du byter styrkort. Detaljerade instruktioner med bilder för att montera styrkortet finns på hemsidan.

5.2 Mjukvara

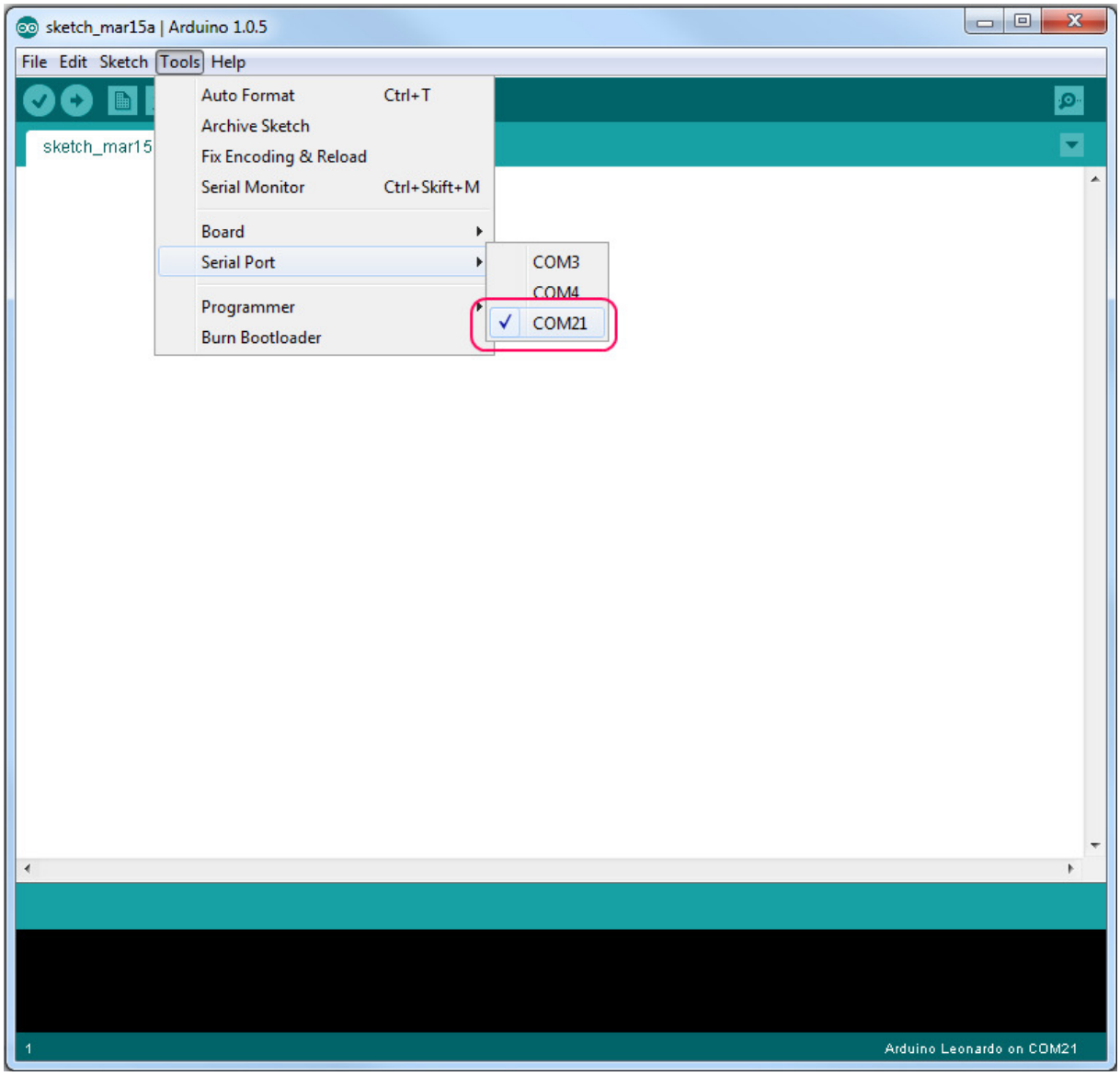
Eftersom Dustino är kompatibelt med Arduino Leonardo följer du enklast samma instruktioner som för Leonardo för att få igång din robot. Börja med att ladda ner utvecklingsmiljön från Arduinos hemsida. Du behöver även ladda hem och installera USB drivrutinerna. Drivrutiner och instruktioner för att installera detta hittar du här.

Anslut sedan Dustino med en USB kabel (typ B, används oftast t.ex. till skrivare och skanners) till din dator och starta Dustino genom att slå på strömmen. Om installationen av USB drivrutinerna fungerade kommer datorn att hitta roboten.

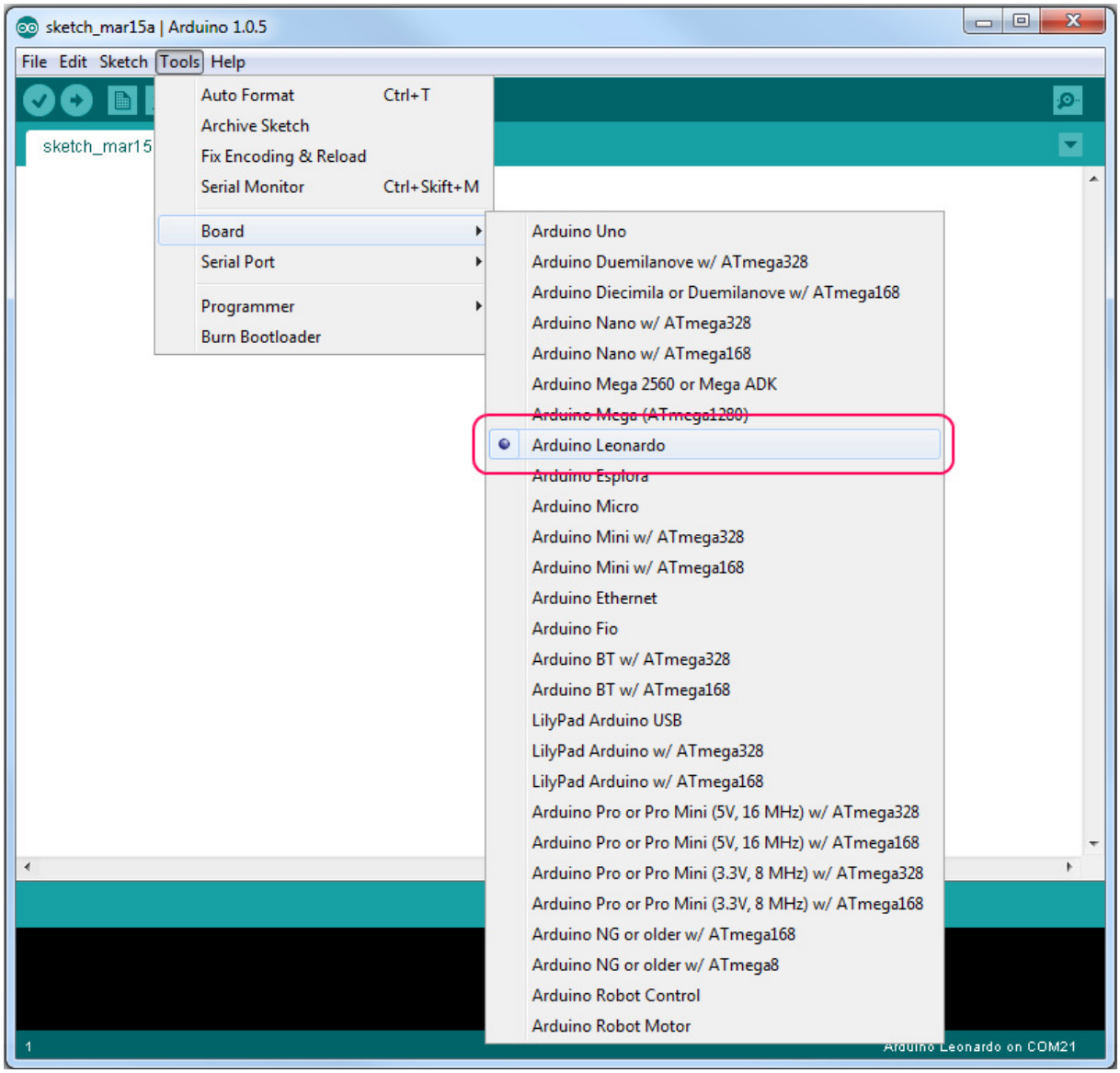
Kontrollera vilken serieport som Dustino använder. Kör du operativsystemet Windows hittar du informationen i Enhetshanteraren.



Starta därefter Arduinos utvecklingsmiljön (förkortat hädanefter IDE, Integrated Development Enviroment) på din dator och markera den aktuella serieporten under menyn *Tools/Serial port*.



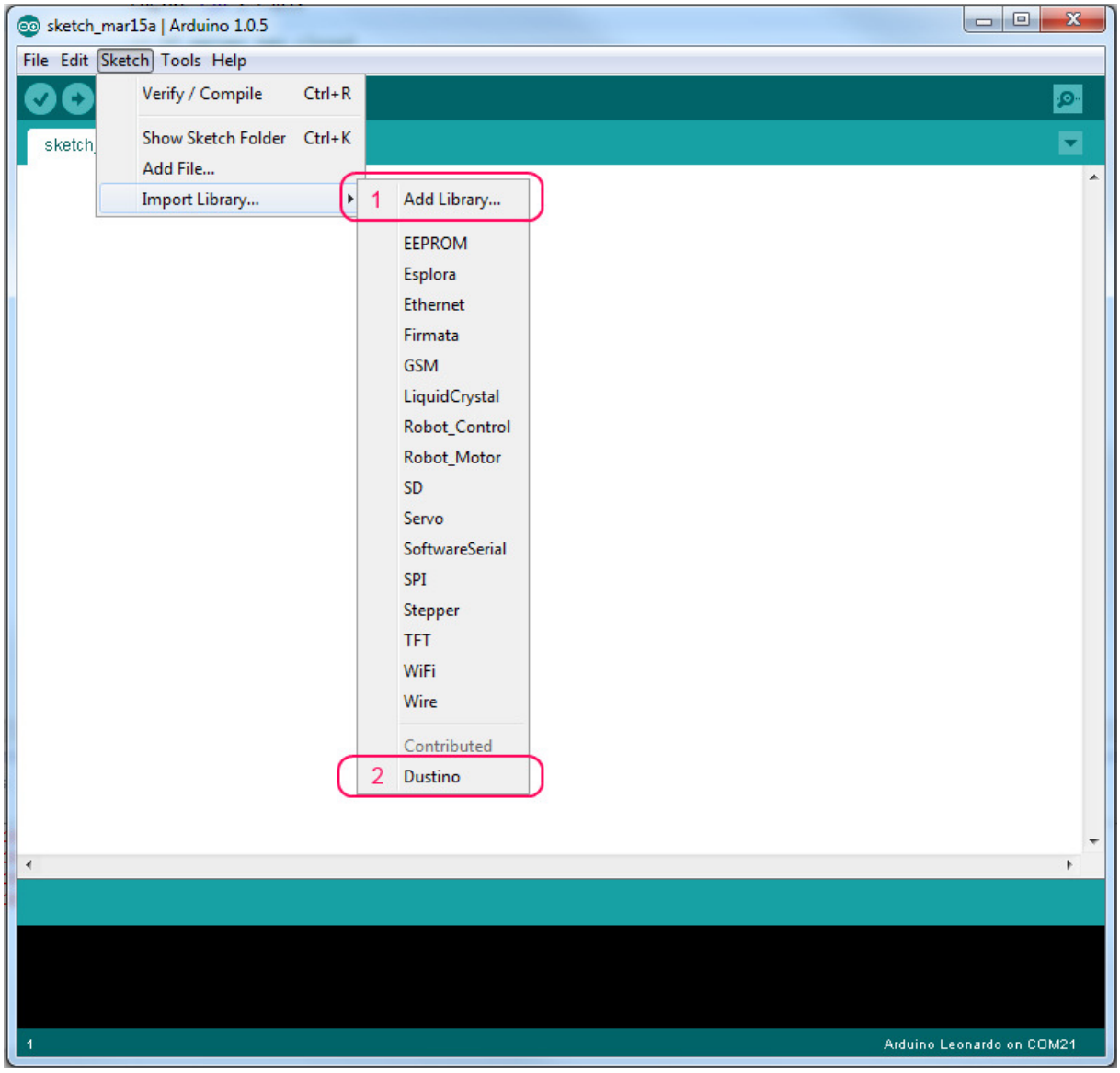
Markera sedan att du använder Arduino Leonardo under *Tools/Board*.



Därmed är Arduino IDE konfigurerat för att du ska kunna skriva och ladda program till din robotdammsugare.

För att du snabbt ska komma igång finns det ett speciellt Dustino-bibliotek som innehåller färdiga metoder att kontrollera din Dustino. Det är rekommenderat att du använder biblioteket när du utvecklar då det innebär att dina program även blir kompatibla med framtida versioner av Dustino. Biblioteket kan du ladda ner härifrån. Klicka på *Sketch/Import library/Add library* i Arduino IDE:et

(nummer 1 i bilden nedan). Välj därefter den mapp där du sparade biblioteket.



Om Arduino lyckas importera biblioteket kommer biblioteket att listas i menyn med de övriga valbara biblioteken. Sista steg är att tala om för Arduino att du vill använda dig av bibliotek i ditt program. Detta gör du genom att klicka på biblioteket (nummer 2).

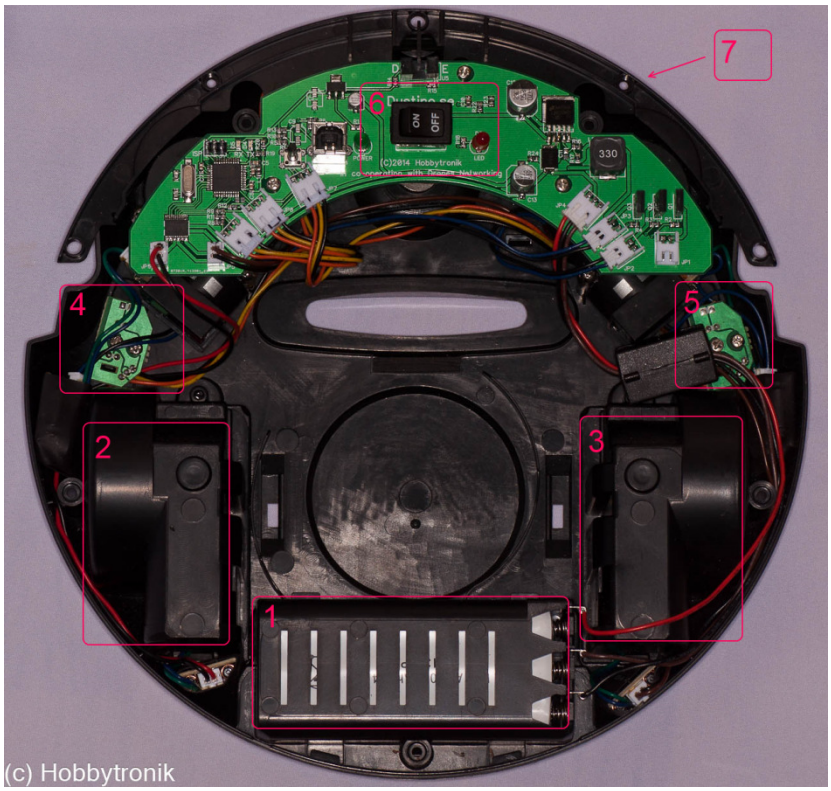
En komplett beskrivning av biblioteket med alla dess metoder hittar du i kapitel *9 Appendix, API* och i kapitel *7 Programmering* hittar du en steg-för-steg guide hur du kan använda de olika metoderna för att få en komplett dammsugarrobot.

6 Roboten

Här följer en kort beskrivning av de delar roboten består av.

6.1 Inuti

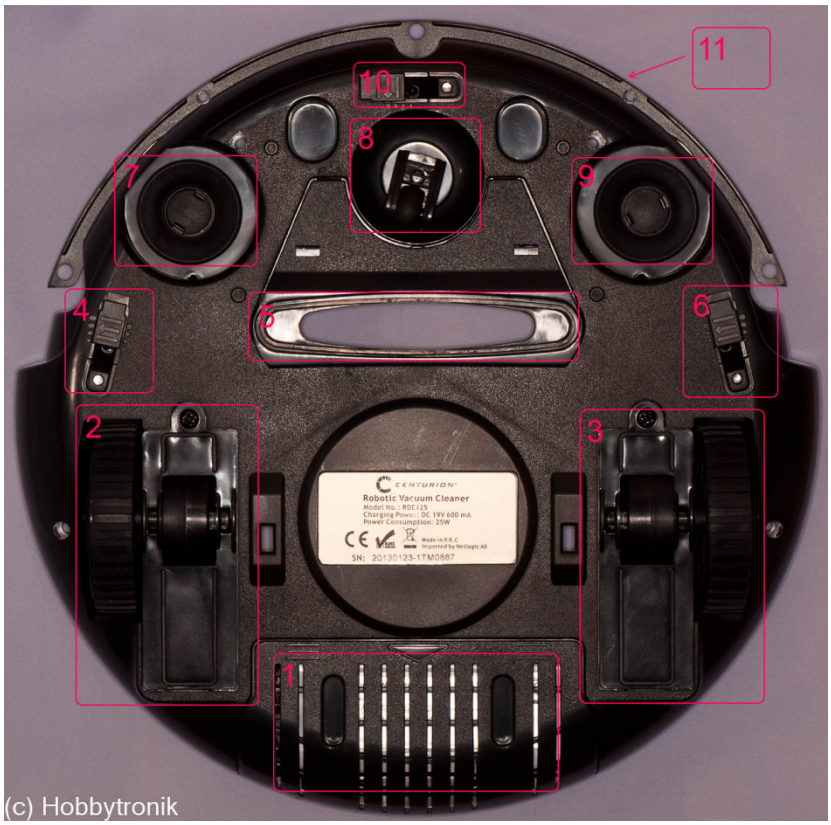
Det mest utmärkande inuti roboten är dess styrkort. Alla delar av roboten som kräver någon form av elektriskt signal kontakteras mot styrkortet.



1. Batteri
2. Vänster hjulmotor
3. Höger hjulmotor
4. Vänster trappsensor
5. Höger trappsensor
6. Styrkortet (se kapitel *6.4 Styrkortet* för detaljer)
7. Stötfångare

6.2 Undersidan

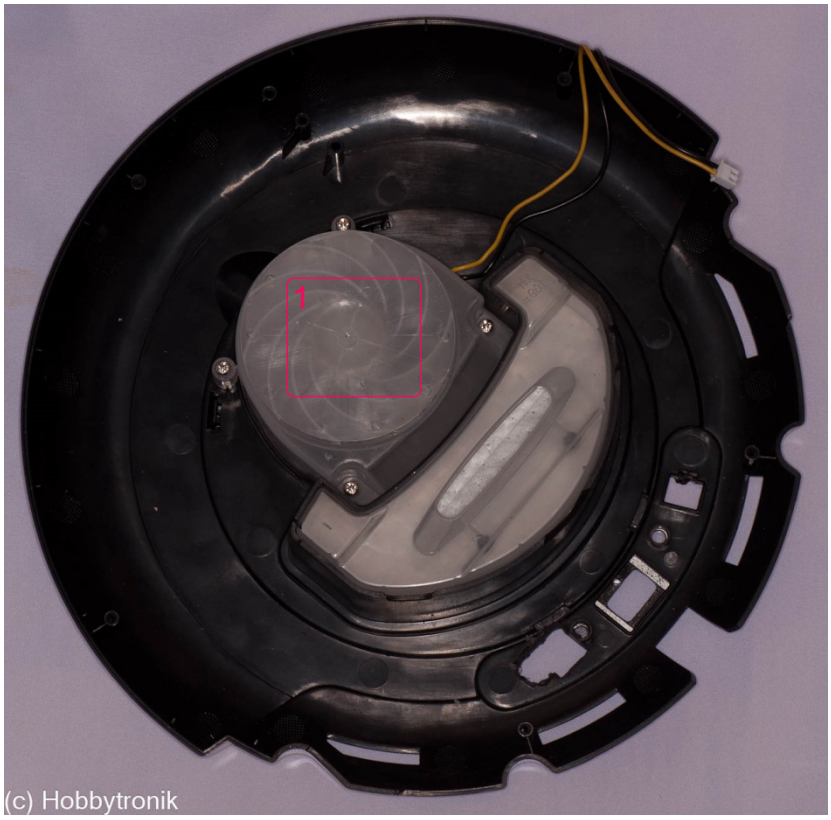
På undersidan av roboten sitter många viktiga delar på roboten. Där sitter det bland annat trappsensorer, hjul, dammintag samt sidoborstarna. Här finns även batteriluckan och ett stödhjul för att roboten ska kunna hålla balansen.



1. Batterilucka
2. Höger hjul
3. Vänster hjul
4. Höger trappsensor
5. Dammintag
6. Vänster trappsensor
7. Höger sidoborste
8. Stödhjul
9. Vänster sidoborste
10. Mitten trappsensor
11. Stötfångare

6.3 Överdelen

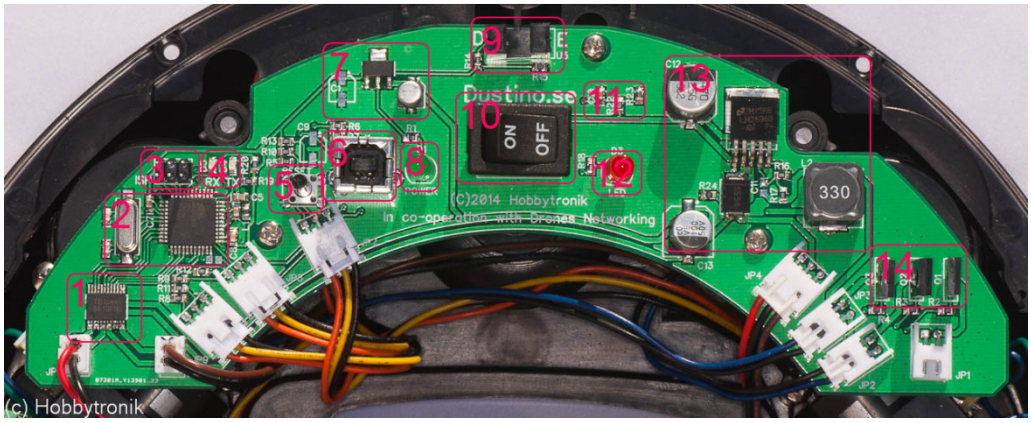
I överdelen av Dustinos chassi sitter dammsugarmotorn. Här finns också manöverpanelen för roboten.



1. Dammsugarmotor

6.4 Styrkortet

Styrkortet kan betraktas som robotens hjärna och kan delas in i ett antal block. Se nedan bild och efterföljande beskrivning av blocken.

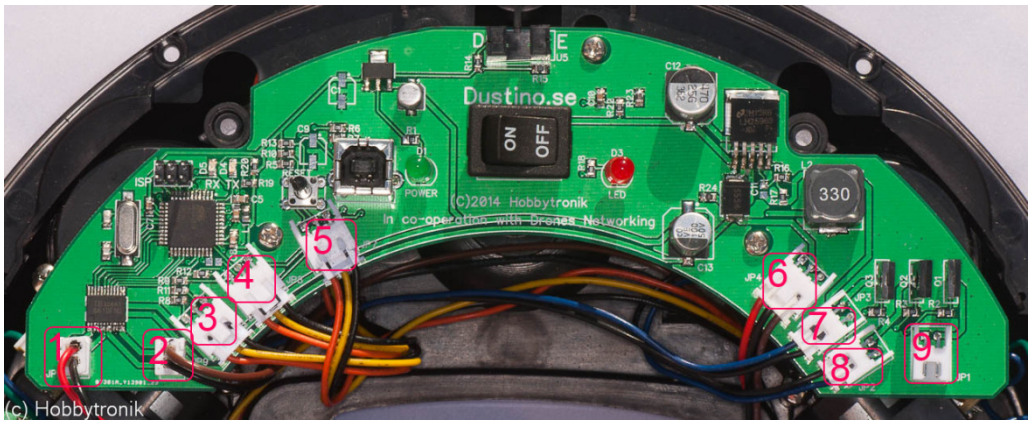


1. Hjulmotordrivare (dubbelt H-brygga)
2. Mikroprocessor med kristall
3. ISP kontakt

4. Indikeringslysdioder
5. Reset-knapp
6. USB anslutning
7. Matning +5 V
8. LED som lyser om roboten är strömsatt
9. Stötsensor
10. Huvudströmbrytare
11. Mätning av batterispänning
12. Programmerbar LED
13. Matning +8 V för motorer
14. Styrning av sidoborstar och dammsugare

6.5 Kontakter

På styrkortet sitter ett antal kontakter där robotens motorer, sensorer och batteri ansluts till. Med undantag för batteriet (nr 6 i bilden nedan) finns det inga krav på att alla delar behöver anslutas. Det kan till och med vara lämpligt att i början inte ansluta t.ex. hjulmotorerna så att roboten inte råkar rusa för att det blivit något fel i mjukvaran. Framförallt bör du vara försiktig med dammsugarmotorn som sitter på överdelen eftersom den snurrar med en hög hastighet och kan i värsta fall orsaka skada. Det är rekommenderat att du bara kör dammsugarmotorn när roboten är fullständigt monterad. Om möjligt, låt därför den kontakten vara oansluten tills du monterar ihop roboten.



1. Vänster hjulmotor
2. Höger hjulmotor
3. Vänster trappsensor
4. Mitten trappsensor
5. Höger trappsensor
6. Batterianslutning
7. Vänster sidoborste
8. Höger sidoborste
9. Dammsugarmotor

7 Programmering

7.1 Inledning

Detta kapitel beskriver Dustino biblioteket i detalj. Biblioteket gör att du inte behöver sätta dig in i alla hårdvaru-detaljer för att kunna använda Dustino utan du kan snabbt få din robot att börja agera med omgivningen. Vidare har biblioteket också fördelen att du kan använda dina program i framtida version av Dustino.

Det förutsätts att du är lite erfaren när det gäller programmering i Arduino. Om du känner dig osäker använd gärna några av de många guider som finns på nätet och som i mer detalj beskriver språket som används av Arduino. Du kan även titta på de färdiga exempel som finns, både de som medföljer i utvecklingsmiljön eller de som du kan ladda ner från enklare-robotar.nu som är speciellt skrivna för Dustino.

Första steget i att börja använda biblioteket är att dels tala om för Arduino att biblioteket ska användas (se tidigare kapitel) och sedan anropa funktionen `init()` för att biblioteket ska kunna konfigurera upp portar och pinnar som den använder sig av. Själva anropet till `init()` görs med fördel i `setup()` metoden.

Exempel

```
#include <Dustino.h>
Dustino dustino;

void setup()
{
     dustino.init();
}
void loop()
{
}
```

Nu kan du börja använda alla metoder som finns i biblioteket. En komplett lista över alla metoder med beskrivning hittar du i kapitel *9 Appendix, API*. I en del exempel framöver anges inte alltid funktionerna `setup()` och `loop()`. Det förväntas vara underförstått att sådana funktioner måste finnas.

Nedan följer ett exempel på hur du kan få lysdioden på roboten att lysa varannan sekund.

Exempel

```
void loop()
{
  dustino.setLED(true);
  delay(1000); // Vänta här i 1 sekund (1000 millisekunder)
  dustino.setLED(false);
  delay(1000);
}
```

7.2 Styra hjulen

Dustino har två hjul som är kopplade till var sin motor. Varje motor kan styras steglöst och individuellt. Genom att variera hur mycket varje hjul ska rotera kan du få roboten att antingen svänga, köra framåt och bakåt eller snurra runt på plats.

Observera

Innan du kör roboten kontrollera att du har tagit bort USB-kabeln eller att roboten står på något som gör att hjulen inte kommer i kontakt med marken annars riskerar du att skada din robot. Var också beredd att lyfta bort roboten så att den inte krockar med något föremål på marken.

Hjulens hastighet kan du bestämma genom att anropa metoderna `setMotorSpeedLeft` eller `setMotorSpeedRight`. Metoderna anropar du med en parameter som ska innehålla ett värde mellan -400 och 400. Värdet bestämmer hur snabbt hjulet ska snurra. Ju högre nummer desto snabbare snurrar hjulet. Ett negativt nummer innebär att hjulet snurrar bakåt. Om du skickar in värdet 0 stannar hjulet. Normalt brukar du vilja ändra på båda motorerna samtidigt och då kan du istället använda metoden `setMotorSpeeds`. Den metoden ska ha två parametrar, en för vänster hjul och en för höger hjul.

Exempel

```
// Roboten åker framåt i halvfart
dustino.setMotorSpeeds(200, 200);
//  Roboten åker framåt i full fart
dustino.setMotorSpeeds(400, 400);
// Roboten snurrar på plats
dustino.setMotorSpeeds(200, -200);
// Roboten åker långsamt bakåt
dustino.setMotorSpeeds(-100, -100);
// Roboten svänger åt höger
dustino.setMotorSpeeds(200, 0);
// Roboten stannar
dustino.setMotorSpeeds(0, 0);
```

Observera

För din säkerhets skull är motorerna till hjulen alltid i viloläge när roboten startar. Det innebär att oavsett vilka parametrar du använder för att sätta motorhastigheten kommer inte motorerna inte att spinna. Det du behöver göra är att anropa funktionen `setStandbyMode(false)` för att ta motorerna ur viloläget.

7.3 Styra övriga motorer

Utöver motorerna till hjulen har Dustino ytterligare 3 motorer. Två av dem är kopplade till sidoborstarna och den tredje är själva dammsugarmotorn.

Sidoborstarna startar du genom att anropa metoden `setLeftSidebrush` respektive `setRightSidebrush`. Anropa metoderna med parametern `true`. Om du istället anger parametern `false` stängs motorerna av.

Exempel

```
// Starta båda sidoborstarna
dustino.setLeftSidebrush(true);
dustino.setRightSidebrush(true);
// Stoppa båda sidoborstarna
dustino.setLeftSidebrush(false);
dustino.setRightSidebrush(false);
```

Dammsugarmotorn startar du genom att anropa `setVaccumMotor` med parametern `true`. Precis som för sidoborstarna stänger du av motorn genom att ange parametern `false`.

Exempel

```
// Starta dammsugarmotorn
dustino.setVaccumMotor(true);
// Stoppa dammsugarmotorn
dustino.setVaccumMotor(false);
```

Observera

För din säkerhets skull bör du aldrig starta dammsugarmotorn om inte roboten är komplett monterad. Motorn snurrar med hög fart och kan orsaka skada vid oförsiktighet.

7.4 Läsa av sensorer

För att Dustino ska kunna agera med sin omgivning är den utrustad med ett antal olika typer av sensorer. Sensorernas jobb är att översätta verkligheten till något som är mätbart för roboten.

Den första typen av sensor på roboten är den så kallade stötsensorn. Den aktiveras när roboten stöter i något och används främst för att få roboten att stanna och byta riktning. Genom att anropa metoden `isBumperSensorActive` kan du läsa av krocksensorn. Metoden returnerar `true` om den har stött på något och `false` annars.

Exempel

```
// Stanna motorerna och tänd lysdioden vid krock
if(dustino.isBumperSensorActive())
{
     dustino.setLED(true);
     dustino.setMotorSpeeds(0, 0);
}
```

Den andra typen av sensorer på roboten är trappsensorerna. Dessa används för att upptäcka om roboten kör ut över en kant t.ex. i anslutning till en trappa eller ett bord. Sensorerna kan även användas för att detektera om roboten lyfts upp.

Det sitter tre stycken trappsensorer under roboten. Två är placerade strax framför vänster respektive höger hjul. Den tredje är placerat i mitten längst fram på roboten. Sensorerna läsas av genom att anropa metoderna `readLeftStairSensor`, `readCenterStairSensor` och `readRightStairSensor`. Metoderna returnerar ett värde mellan 0 och 1023. Ett högt värde (större än 1020) innebär att roboten inte kan detektera golv under sensor.

Exempel

```
// Stanna roboten om roboten har lyfts upp
if((dustino.readLeftStairSensor() > 1020) &&
   (dustino.readCenterStairSensor() > 1020) &&
       (dustino.readRightStairSensor() > 1020))
{
     dustino.setMotorSpeeds(0, 0);
}
```

Den tredje typen av sensor som finns på roboten är batterinivå-sensorn. Den läser av hur mycket spänning det finns i batteriet. När batteriet är fullt ligger spänningen på ca 18 V och när det är dags att ladda ca 10.8 V. Spänning bör aldrig understiga 10.8 V för att minimera riskerna att skada batteriet permanent. Använd metoden `readBatteryLevel()` som returnerar aktuell spänning.

Exempel

```
// Om spänning är för låg, stäng av motorerna och tänd lysdioden
if( dustino.readBatteryLevel() < 11.0f )
{
 dustino.setLeftSidebrush(false);
 dustino.setRightSidebrush(false);
 dustino.setVaccumMotor(false);
 dustino.setStandbyMode(true);

 dustino.setLED(true);

 while(1);

}
```

7.5 En mer komplett program

Att testa enstaka funktioner är ett utmärkt sätt att lära sig om vilka möjligheter som finns med roboten men det är först när man kombinerar flera funktioner efter varandra på ett smart sätt som man kan börja prata om intelligens hos en robot.

När du börjar programmera är det bra om du redan från början har en idé om vad du vill att din robot ska göra. Målet med denna övning är att få en komplett fungerande dammsugarrobot. Mycket av konsten att bli en bra programmerare handlar om att kunna bryta ner mål i mindre och därmed mer lätthanterliga delar. Vi bryter därför ner målet i flera delmål:

1) Dammsuga (så klart!)
2) Köra slumpmässigt runt i rummet
3) Detektera ev. hinder på vägen eller väggar
4) Kontrollera att batterinivån inte är för låg
5) Kontrollera att roboten inte riskera att falla t.ex. vid en trappa
6) Stänga av alla motorer då den lyfts

Starta en ny *Sketch* och utgå från följande exempel

Exempel

```
Dustino dustino;

void setup()
{
     dustino.init();
}
void loop()
{
}
```

Kontrollera att *sketchen* kan kompileras och att den laddas ner till roboten utan problem.

Nästa steg är att börja röra roboten. Det är viktigt att komma ihåg att anropa funktionen `setStandbyMode` för att ta roboten ur sitt viloläge så att vi kan starta motorerna.

Exempel

```
Dustino dustino;

void setup()
{
     dustino.init();
     dustino.setStandbyMode(false);
}

void loop()
{
     dustino.setMotorsSpeed(200, 200);
}
```

Vi väljer att låta roboten köra rakt fram på halvfart. Ett bra tips är att alltid låta roboten köra långsamt i början av utvecklingen. Dels hinner du reagera om roboten börjar köra åt fel håll och dels minskar risken för allvarliga skador på människa, djur och ting.

Nästa mål är att kontrollera så att roboten inte kör in i något. Detta kan göras med funktionen `isBumperSensorActive()`. Funktionen returnerar `true` om roboten kört på något och vi kan då utnyttja detta till att stänga av motorerna till hjulen.

Exempel

```
void loop()
{
     // Kontrollera om roboten har kört på något
     if(dustino.isBumperSensorActive())
     {
          // Jepp, stäng av motorerna!
          dustino.setMotorsSpeed(0, 0);
     }
     else
     {
```

```
            // Nej, ingen fara, Roboten kan fortsätta rakt fram
            dustino.setMotorsSpeed(200, 200);
        }
}
```

En dammsugarrobot som kör rakt fram och stannar när den kör på något har man ingen större nytta av. Nästa steg är att låta den svänga för att undvika hindret framför den.

Exempel

```
Dustino dustino;

void setup()
{
     dustino.init();
     dustino.setStandbyMode(false);

     // Skapa en slumptalsbörjan genom att läsa av en
     // analog ingång som inte är kopplat till något
     RandomSeed(analogRead(A9));
}

void loop()
{
     // Kontrollera om roboten har kört på något
     if(dustino.isBumperSensorActive())
     {
          dustino.setLED(true);

          // Jepp, stäng av motorerna!
          dustino.setMotorsSpeed(0, 0);

          // Backa först från hindret så att roboten kan
          // svänga obehindrat
          dustino.setMotorsSpeed(-100, -100);
          delay(200);

          // Slump fram åt vilket håll roboten ska svänga
          If(Random(0, 1) == 0)
               dustino.setMotorsSpeed(-150, 150);
          else
               dustino.setMotorsSpeed(150, -150);

          // Slumpa hur länge roboten ska svänga
          delay(Random(200, 400));

          dustino.setMotorsSpeed(0, 0);
     }
     else
     {
```

```
            // Nej, ingen fara, Roboten kan fortsätta rakt fram
            dustino.setMotorsSpeed(200, 200);
            dustino.setLED(false);
      }
}
```

Vi passar också på att tända den röda lysdioden vid en krock. När vi väl har kommit så här långt har vi faktiskt fått ihop en robot som slumpmässigt åker runt i rummet och undviker att krocka med hinder. Notera att vi har ett antal `delay` i programmet för att få roboten att utföra aktiviteter under en viss tid. Dessa tider behöver justeras för att få en bra funktion.

Slår vi på sidoborstarna och dammsugaren har vi fått oss en alldeles egen dammsugarrobot!

Exempel

```
Dustino dustino;

void setup()
{
      dustino.init();
      dustino.setStandbyMode(false);

      // Skapa en slumptalsbörjan genom att läsa av en
      // analog ingång som inte är kopplat till något
      RandomSeed(analogRead(A9));

      // Aktivera sidoborstarna och dammsugarmotorn
      dustino.setLeftSidebrush(true);
      dustino.setRightSidebrush(true);
      dustino.setVaccumMotor(true);
}
```

När de grundläggande funktionerna är klara är det dags att kika på de mer säkerhetsmässiga. En sak som vi behöver vara rädd om är batteriet. Vi får inte låta roboten helt ladda ur batteriet vilket är lätt hänt när man har stora laster som t.ex. motorer. Ett urladdat batteri kan nämligen förstöras och blir omöjligt att laddas upp igen.

Exempel

```
Dustino dustino;

void setup()
{
      dustino.init();
      dustino.setStandbyMode(false);

      // Skapa en slumptalsbörjan genom att läsa av en analog
      // ingång som inte är kopplat till något
      RandomSeed(analogRead(A9));

      // Aktivera sidoborstarna och dammsugarmotorn
```

```
	dustino.setLeftSidebrush(true);
	dustino.setRightSidebrush(true);
	dustino.setVaccumMotor(true);
}

void loop()
{
	// Kontrollera att vi inte når en kritiskt batterinivå
	if(dustino.readBatteryLevel() < 11.0f)
	{
		// Stäng av alla motorer
		dustino.setLeftSidebrush(false);
		dustino.setRightSidebrush(false);
		dustino.setVaccumMotor(false);
		dustino.setMotorsSpeed(0 ,0 );

		// Blinka snabbt så länge batterinivån är låg
		while(dustino.readBatteryLevel() < 11.0f)
		{
			dustino.setLED(true);
			delay(200);
			dustino.setLED(false);
			delay(200);
		}
	}
	// Kontrollera om roboten har kört på något
	if(dustino.isBumperSensorActive())
	{
		dustino.setLED(true);

		// Jepp, stäng av motorerna!
		dustino.setMotorsSpeed(0, 0);

		// Backa först från hindret så att roboten kan
		// svänga obehindrat
		dustino.setMotorsSpeed(-200, -200);
		delay(200);

		// Slump fram åt vilket håll roboten ska svänga
		if(random(0, 1) == 0)
			dustino.setMotorsSpeed(-150, 150);
		else
			dustino.setMotorsSpeed(150, -150);

		// Slumpa hur länge roboten ska svänga
		delay(Random(200, 400));
		dustino.setMotorsSpeed(0, 0);
	}
	else
```

```
        {
            // Nej, ingen fara, Roboten kan fortsätta rakt fram
            dustino.setMotorsSpeed(200, 200);
            dustino.setLED(false);
        }
    }
```

Ett annan delmål vi har med denna övning är att stänga av alla motorer om roboten lyfts. Anledningen är förstås att roboten blir mer lätthanterlig. Genom att lyfta upp roboten kan vi snabbt hindra roboten från eventuella missöden. Ett tricks för att roboten ska veta om den har lyfts upp eller inte är att kontrollera trappsensorerna. Om alla trappsensorer varnar för trappa kan vi gissa oss till att roboten antagligen har lyfts upp. Vi kan förstås inte vara helt säkra. Trappsensorerna skulle t.ex. varna om roboten befinner sig i fritt fall men i vilket fall som helst bör motorerna stängas av.

```
// Kontrollera trappfunktionen
  // Om alla sensorer indikerar att det inte finns något golv under
roboten => Roboten är upplyft, stäng av alla motorer
  if( dustino.readLeftStairSensor() > 1010 &&
      dustino.readCenterStairSensor() > 1010 &&
      dustino.readRightStairSensor() > 1010 )
  {
      stopMotors();

      // Blinka med LED:en så länge som roboten hålls lyft
      while( dustino.readLeftStairSensor() > 1010 ||
             dustino.readCenterStairSensor() > 1010 ||
             dustino.readRightStairSensor() > 1010 )
      {
        dustino.setLED(true);
        delay(200);
        dustino.setLED(false);
        delay(200);
      }

      startMotors();
  }
```

Sista delmålet är att få roboten att reagera på trappor och avsatser. Detta görs på liknande sätt som att detektera om roboten lyfts men istället kontrollerar vi de enskilda trappsensorerna.

```
// Kontrollera trappfunktionen
  // Om alla sensorer indikerar att det inte finns något golv under
roboten => Roboten är upplyft, stäng av alla motorer
  if( dustino.readLeftStairSensor() > 1010 &&
      dustino.readCenterStairSensor() > 1010 &&
      dustino.readRightStairSensor() > 1010 )
  {
      stopMotors();

      while( dustino.readLeftStairSensor() > 1010 ||
```

```
            dustino.readCenterStairSensor() > 1010 ||
            dustino.readRightStairSensor() > 1010 )
      {
        dustino.setLED(true);
        delay(200);
        dustino.setLED(false);
        delay(200);
      }

      startMotors();
  }
// Vänster sensor indikerar trappa, sväng roboten åt höger
  else if(dustino.readLeftStairSensor() > 1010)
  {
    turn(1);
  }
// Högre sensor indikerar trappa, sväng roboten åt vänster
  else if (dustino.readCenterStairSensor() > 1010)
  {
    turn(0);
  }
// Mitten sensor indikerar trappa, sväng roboten slumpmässigt åt höger
// eller vänster
  else if (dustino.readRightStairSensor() > 1010)
  {
    turn(random(0, 1));
  }
```

Observera anropet av funktionen `turn`. Behovet att svänga roboten år olika håll finns på flera ställen i programmet och för att undvika att vi skriver samma sak flera gånger bryter vi hellre ut det till en egen funktion.

```
void turn(int dir)
{
  // Jepp, stäng av motorerna!
  dustino.setMotorSpeeds(0, 0);
  delay(200);

  // Backa först från hindret så att roboten kan
  // svänga obehindrat
  dustino.setMotorSpeeds(-200, -200);
  delay(1200);

  // Kontrollera vilket håll roboten ska snurra
  if(dir == 0)
    dustino.setMotorSpeeds(-150, 150);
  else
    dustino.setMotorSpeeds(150, -150);
```

```
    // Slumpa hur länge roboten ska svänga
    delay(random(800, 1200));
    dustino.setMotorSpeeds(0, 0);
}
```

7.6 Förslag på förbättringar

Även den sketch som vi har tagit fram i förgående kapitel är fullt tillräckligt för att nå vårt mål med övningen, att få en fungerande robotdammsugare finns det fortfarande saker som vi kan förbättra och bygga vidare på. När du har kommit så här långt har du säkert redan själv idéer och tankar på nya uppgifter för din robot men här följer ytterligare några förslag på vad du kan förbättra:

- När roboten slås på drar motorerna igång direkt. Låt den istället vänta några sekunder så att du hinner placera roboten och ta bort händerna innan den startar.
- Strukturera upp koden genom att bryta ut delar i separata metoder. Detta bidrar inte bara till att din kod blir mer lättläst, du kan även återvinna kod och minska antal fel.
- Används konstanter istället för hårdkodade värden.
- Vår robot rör sig slumpmässigt i rummet. Testa att få den t.ex. att åka runt i en spiral eller andra mönster för att hitta det bästa sättet att snabbt städa ett rum.
- Det är fullt möjligt att skriva metoder för att varvtalsstyra både sidoborstarna och dammsugarmotorn. Det kräver dock en djupare förståelse av hur Arduino fungerar under huven och rekommenderas för dig som är lite mer erfaren.
- Gör en linjeföljande robot genom att utnyttja trappsensorerna.
- Hanteringen av hur länge roboten ska t.ex. snurra beror dels på tiden men också hastigheten på hjulen. Gör denna hantering bättre genom att införa en funktion vars ingående parametrar är hur många grader roboten ska snurra och hur snabbt hjulmotorerna ska snurra.

8 Appendix, Länkar

Här följer några länkar som är bra att känna till:

`www.enklare-robotar.nu` - Företaget bakom Dustino

`www.dustino.se` - Robotens hemsida, här hittar du t.ex. exempel, schema, kursmaterial och mycket mer som hör till roboten

`www.arduino.cc` - Arduinos hemsida

`http://arduino.cc/en/Main/Software` - Länk för nerladdning av Arduino IDE

`http://arduino.cc/en/Main/ArduinoBoardLeonardo` - Arduino Leonardos hemsida

`http://arduino.cc/en/Guide/ArduinoLeonardoMicro` - Guide för Arduino Leonardo

`http://www.atmel.com/Images/doc7766.pdf` - Länk till databladet för Atmel AVR ATMega32U4

`http://www.netlogic.se/rde125.aspx` - Information om dammsugarroboten

9 Appendix, API

Detta kapitel beskriver de funktioner som finns i biblioteket Dustino. Biblioteket innehåller ett antal färdiga funktioner för att snabbt kunna komma igång med programmeringen av roboten.

9.1 void init()

Denna funktion måste anropas först innan några andra funktioner får lov att anropas. Funktionen initierar och konfigurerar de in- och utgångar som Dustino använder sig av.

Exempel

```
Dustino dustino;

// the setup routine runs once when you press reset:
void setup() {
   // Här får man ej anropa några funktioner i Dustino biblioteket

   dustino.init();

  // Här är det fritt fram att anropa funktioner
}

// the loop routine runs over and over again forever:
void loop() {
   // Eftersom dustino.init() anropades i setup() är det tillåtet att
      anropa andra funktioner här
}
```

9.2 void setLED(boolean active)

Denna metod används för att styra den röda lysdioden i roboten. Lysdioden kan t.ex. användas för att indikera ett fel eller ge ett enklare status-meddelande.

Exempel

```
void loop() {
   dustino.setLED(true);    // Tänder lysdioden
   delay(200);  // Väntar en stund
   dustino.setLED(false);   // Släcker lysdioden
   delay(200);  // Väntar en stund
}
```

9.3 boolean isLEDActive()

Returnerar `true` ifall den röda lysdioden lyser. Annars returnerar funktionen `false`.

Exempel

```
void loop() {
   if(dustino.isLEDActive())  // Kontrollerar om lysdioden lyser
      delay(1000);  // Ja, det gör den, vänta extra länga
   else
      delay(100);  // Nej, lysdioden lyser inte, vänta en kort stund
}
```

9.4 void setLeftMotorSpeed(int speed)

Bestämmer hastigheten på vänstra hjulet. Hastigheten kan sättas mellan -400 och 400 där -400 betyder maximal fart bakåt och 400 maximal fart framåt. Om hastigheten sätts till 0 stannar hjulet.

Exempel

```
void loop() {
  setLeftMotorSpeed(400); // Maximal framåt med vänster hjul.
  delay(500);  // Vänta 500 ms

  setLeftMotorSpeed(0);    // Stanna vänstra hjulet.
  delay(500);  // Vänta 500 ms

  setLeftMotorSpeed(-400); // Kör vänstra hjulet i maximal hastighet
                           bakåt
  delay( 500);  // Vänta 500 ms

  setLeftMotorSpeed(0);    // Stanna vänstra hjulet.
}
```

9.5 void setRightMotorSpeed(int speed)

Bestämmer hastigheten på högra hjulet. Hastigheten kan sättas mellan -400 och 400 där -400 betyder maximal fart bakåt och 400 maximal fart framåt. Om hastigheten sätts till 0 stannar hjulet.

Exempel

```
void loop() {
  setRightMotorSpeed(400); // Maximal framåt med höger hjul.
  delay(500);  // Vänta 500 ms

  setRightMotorSpeed(0);   // Stanna högra hjulet.
  delay(500);  // Vänta 500 ms

  setRightMotorSpeed(-400);  // Kör högra hjulet i maximal
                             hastighet bakåt
  delay(500);  // Vänta 500 ms

  setRightMotorSpeed(0);   // Stanna högra hjulet.
}
```

9.6 void setMotorSpeeds(int leftMotorSpeed, int rightMotorSpeed)

Bestämmer hastigheten på både hjulen samtidigt. Hastigheten kan sättas oberoende av varandra. Hastigheten kan sättas mellan -400 och 400 där -400 betyder maximal fart bakåt och 400 maximal fart framåt. Om hastigheten sätts till 0 stannar hjulet.

Exempel

```
void loop() {
  setMotorSpeeds(400, 400);           // Maximal framåt med båda hjulen
  delay(500);  // Vänta 500 ms
  setMotorSpeeds(200, 0); // Snurra roboten
  delay(500);  // Vänta 500 ms
}
```

9.7 void setStandbyMode(boolean active)

Denna funktion kan användas för att sätta hjulmotorerna i viloläge. Viloläget innebär att motorerna inte kommer att spinna oavsett vilken hastighet motorerna ställs in för. Roboten startar alltid med motorerna i viloläge. Du måste aktivt anropa denna funktion med parameterna `false` för att kunna köra roboten.

Exempel

```
void setup() {
   dustino.init();
   dustino.setStandbyMode(false);
}

void loop() {
   // Eftersom viloläget är avstängt i setup kan vi sätta
   // hastigheten på hjulmotorerna
   setMotorsSpeed(200, 200);
}
```

9.8 boolean isInStandbyMode()

Returnerar `true` om motorerna är i viloläge, annars `true`.

Exempel

```
void loop() {
   // Aktivera lysdioden om motorerna är i viloläge
   dustino.setLED(dustino.isInStandbyMode());
}
```

9.9 void setLeftSidebrush(boolean active)

Denna funktion används för att starta eller stanna vänstra sidoborsten. Om parametern är satt till `true` kommer sidoborsten att börja sopa och om den är `false` kommer den att stanna.

Exempel

```
void loop() {
   dustino.setLeftSidebrush(true);  // Starta sidobortsen
   delay(1000); // Dammsug i 1 sekund
   dustino.setLeftSidebrush(false);  // Stäng av sidobortsen
}
```

9.10 boolean isLeftSidebrushActive()

Returnerar `true` om den vänstra sidoborsten snurrar, annars `false`.

Exempel

```
void loop() {
    // Tänd lysdioden om vänstra sidoborsten snurrar
   dustino.setLED(dustino.isLeftSidebrushActive());
}
```

9.11 void setRightSidebrush(boolean active)

Denna funktion används för att starta eller stanna högra sidoborsten. Om parametern är `true` kommer sidoborsten att börja sopa och om den är satt till `false` kommer den att stanna.

Exempel

```
void loop() {
  dustino.setRightSidebrush(true);  // Starta sidobortsen
  delay(1000); // Dammsug i 1 sekund
  dustino.setRightSidebrush(false);  // Stäng av sidobortsen
}
```

9.12 boolean isRightSidebrushActive()

Returnerar true om den högra sidoborsten snurrar, annars false.

Exempel

```
void loop() {
  // Tänd lysdioden om högra sidoborsten snurrar
  dustino.setLED(dustino.isRightSidebrushActive());
}
```

9.13 void setVaccumMotor(boolean active)

Med denna funktion kan dammsugarmotorn slås av eller på. Om metoden anropas med parametern `true` kommer dammsugarmotorn att starta men om metoden anropas med parametern `false`, stängs dammsugarmotorn av.

Exempel

```
void loop() {
  dustino.setVaccumMotor(true);  // Starta dammsugarmotorn
  delay(1000); // Dammsuga i 1 sekund
  dustino.setVaccumMotor(false);  // Stäng av dammsugarmotorn
}
```

9.14 boolean isVaccumMotorActive()

Returnerar `true` om den högra sidoborsten snurrar, annars `false`.

Exempel

```
void loop() {
  // Tänd lysdioden om dammsugarmotorn går
  dustino.setLED(dustino.isVaccumMotorActive());
}
```

9.15 float readBatteryLevel()

Denna funktion returnerar aktuell batterispänning i volt (V). Batterispänningen bör aldrig understiga 10.8 V för att undvika permanenta skador på batteriet.

Notera att felet på den uppmätta nivån kan skiljas uppemot +-5 % från det faktiska värdet pga. komponent- och batteritoleranser.

Exempel

```
void loop() {
   // Vid låg spänning, tänd lysdioden
   dustino.setLED(dustino.readBatteryLevel() < 11.0f);
}
```

9.16 int readLeftStairSensor()

Funktionen läser av den vänstra trappsensorn (placerad strax framför det vänstra hjulet) och returnerar ett värde mellan 0 och 1023 beroende på hur mycket ljus som reflekterats från golvet. Ett högre värde indikerar större sannolikhet att roboten uppfattar att den kommit fram till en kant. Eftersom andelen ljus som reflekteras beror på golvets yta går det inte att ge några exakta gränslägen men som utgångspunkt kan du räkna med att om sensorn returnerar ett värde större än 1020 har roboten kommit fram till en kant.

Exempel

```
void loop() {
   // Stoppa vänster hjul om trappsensor indikerar att roboten
   // kommit till en kant
   if(dustino.readLeftStairSensor() > 1020 )
   {
      dustino.setLeftMotorSpeed(0);
   }
}
```

9.17 int readCenterStairSensor()

Funktionen läser av trappsensorn i fronten på roboten och returnerar ett värde mellan 0 och 1023 beroende på hur mycket ljus som reflekterats från golvet. Ett högre värde indikerar större sannolikhet att roboten uppfattar att den kommit fram till en kant. Eftersom andelen ljus som reflekteras beror på golvets yta går det inte att ge några exakta gränslägen men som utgångspunkt kan du räkna med att om sensorn returnerar ett värde större än 1020 har roboten kommit fram till en kant.

Exempel

```
void loop() {
   // Stoppa båda hjulen hjul om trappsensor indikerar att roboten
   // kommit till en kant
   if(dustino.readCenterStairSensor() > 1020 )
   {
      dustino.setMotorsSpeed(0, 0);
   }
}
```

9.18 int readRightStairSensor()

Funktionen läser av den högra trappsensorn (placerad strax framför det högra hjulet) och returnerar ett värde mellan 0 och 1023 beroende på hur mycket ljus som reflekterats från golvet. Ett högre värde indikerar större sannolikhet att roboten uppfattar att den kommit fram till en kant. Eftersom andelen ljus som reflekteras beror på golvets yta går det inte att ge några exakta gränslägen men

som utgångspunkt kan du räkna med att om sensorn returnerar ett värde större än 1020 har roboten kommit fram till en kant.

Exempel

```
void loop() {
   // Stoppa höger hjul om trappsensor indikerar att roboten
   // kommit till en kant
   if(dustino.readRightStairSensor() > 1020 )
   {
      dustino.setRightMotorSpeed(0);
   }
}
```

9.19 boolean isBumperSensorActive()

Metoden läser av krocksensorn och returnerar true om stötsensorn indikerar att roboten har kört in i något, annars returnerar metoden false.

Exempel

```
void loop() {
   // Stoppa motorerna och tänd lysdioden
   if(dustino.isBumperSensorActive())
   {
      dustino.setMotorsSpeed(0, 0);
      dustino.setLED(true);
   }
}
```