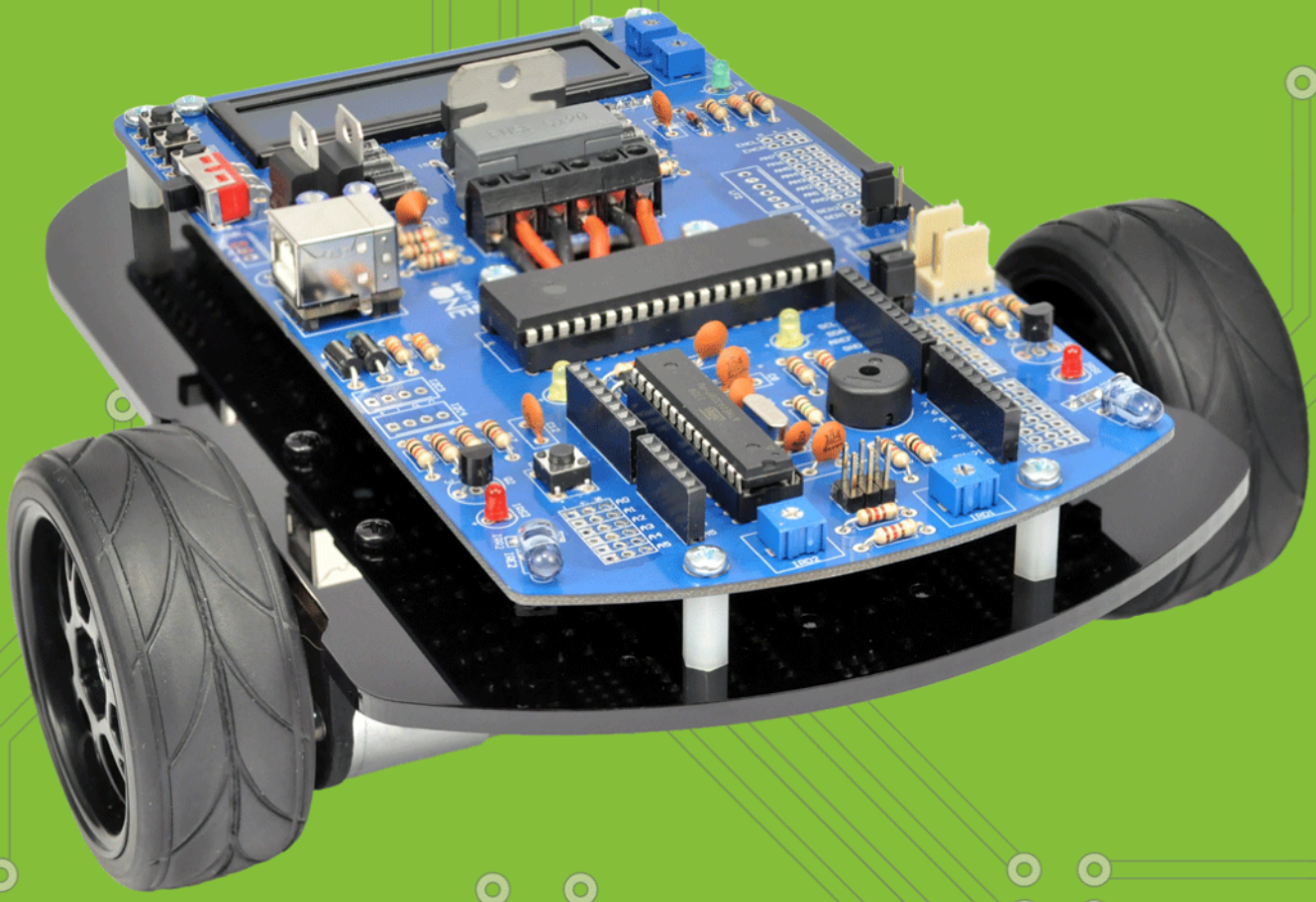
bot'n roll ONE

build your own robot



software brugermanual

www.botnroll.com

CONTENTS

Document Revision: February 15, 2016

1. INTRODUKTION

Bot'n Roll ONE A programmeres i C-sproget i Arduino IDE-miljøet. ATmega328-mikrokontrolleren på denne robot har Arduino Uno bootloader, så robotten programmeres som om den er en Arduino Uno.

Robotten har en anden mikrocontroller, en forprogrammeret PIC18F45K22 med software udviklet af botnroll.com. På Bot'n Roll ONE A fungerer den som en **slave**, der udfører kommandoerne, der er givet af **master** ATmega328.

De to mikrokontrollere på Bot'n Roll ONE A kommunikerer med hinanden gennem SPI-bussen "**Serial Peripheral Interface**". Mikrokontrollerne udveksler information på en koordineret og veldefineret måde. En dataoverførselsprotokol er udviklet for at tillade kommunikation mellem master og salve. Masteren bruger en liste over kommandoer, der svarer til kontrolinstruktioner, og hver kommando genererer et svar fra slaven. Listen over kommandoer og hvordan data overføres mellem master og slave er defineret i BnrOneA-biblioteket.

BnrOneA-biblioteket til Arduino gør det muligt for brugeren at kontrollere robotten på en enkel måde, idet det kun er nødvendigt at bruge bibliotekskommandoerne i Arduino IDE. Disse kommandoer er listet og forklaret i denne manual.

Selvom de to mikrokontrollere kan programmeres på C-sprog, er det kun ATmega328 med Arduino bootloader, der er brug for at programmere ved hjælp af BnrOneA-biblioteket.

PIC18F45K22 kan programmeres på C-sprog ved hjælp af Microchips MPLABX IDE-programmeringsmiljø og XC8-compiler eller anden kompatibel software. Dette bør dog kun gøres af avancerede brugere, da programmering af PIC18F45K22 til at inkludere en ny funktion, kræver, at du også opdaterer BnrOneA-biblioteket til Arduino for at kunne bruge den nye funktionalitet. Kontakt botnroll.com, hvis du ønsker en ny funktion, der skal implementeres på din Bot'n Roll ONE A!

1.1 PROGRAMMERING AF BOT'N ROLL ONE A

For at programmere Bot'n Roll ONE A er det nødvendigt, at du har din computer forberedt med alle nødvendige værktøjer, dvs.

- VCP-driver installeret, Bot'n Roll ONE A USB-portdriver (se BILAG A);
- Arduino IDE installeret (se BILAG B);
- BnrOneA-bibliotek installeret på Arduino IDE (se BILAG B).

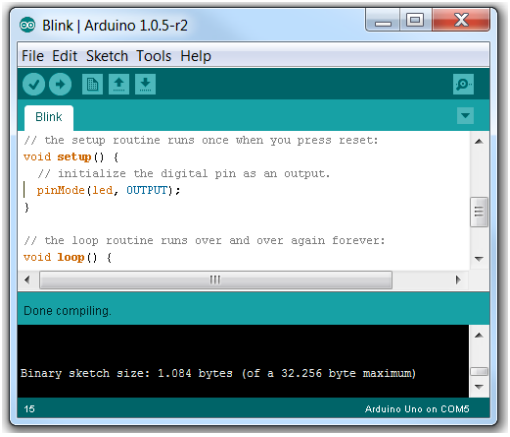
For detaljerede oplysninger om installation af disse værktøjer, se Apendix A og B i enden af denne manual.

C-sproget er også et af de anbefalede værktøjer til programmering af Bot'n Roll ONE A. Hvis du stadig ikke er familiær med C-sproget, kan du finde materiale på biblioteket som en god indføring i begynderprogrammering. RoboParty-præsentationer om C-programmering er også en god inføring, og selvfølgelig er der tusinder af internetwebsteder, der forklarer C-sproget!

1.1.1 Arduino IDE

Arduino-programmeringsmiljøet indeholder en teksteditor til at skrive kode, et område til meddelelser, en tekstkonsol, en værktøjslinje med de vigtigste funktioner og et antal menuer. Det giver også mulighed for at linke til Bot'n Roll ONE A Arduino-hardware for at overføre koden og kommunikere med robotten.

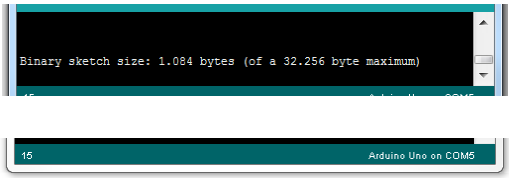
Et program til Arduino kaldes en "**sketch**", der skrives i teksteditoren og gemmes med filtypenavnet ".ino" på din computer.



Meddelelsesområdet viser oplysninger om lagring og eksport af programmer og viser eventuelle fejl.

Konsollen viser tekstbeskeder med detaljerede oplysninger om fejlene og anden type information.

On the bottom right corner of the window the information about the board to be programmed and the serial port in use is presented.



Figur 1: Arduino IDE-moduler

The main icon buttons on the toolbar and its functions are:

***Verificér**:* Tjekker for fejl i koden.

Upload: Kompilerer Compiles the code and sends it to Arduino.

Ny: Laver en ny sketch.

Åbn: Åbner en gemt sketch.

Gem: Gemmer **sketchen**.

Serial Monitor: Åbner en seriel kommunikationsport til en seriel skærm.

Den serielle skærm giver dig mulighed for at se data, der sendes fra Arduino til computeren, og giver dig også mulighed for at sende data fra computeren til Arduino. Det er meget nyttigt i programmeringen, fordi her kan du udskrive værdier for tekst og variabler, og det fungerer således som et "**debugging**" -værktøj til dit program. Når du åbner den serielle skærm, genstarter dit Arduino-program.

1.1.2 BnrOneA Library for Arduino

Et libary er et "forudskrevet" sæt rutiner, som du kan inkludere på din kode og bruge på dit program. For at bruge BnrOneA-biblioteket skal du bare inkludere det i din kode:

#include <BnrOneA.h>

Og oprette en tilgang til klassen:

BnrOneA:BnrOneA;

Fra nu af har du adgang til alle biblioteksfunktioner, der er lavet forud via den forekomst, som du definerede i: **one.library_function ();**

Et bibliotek oprettes normalt til at behandle data eller hardware og har altid mindst to filer, men i Arduino-tilfældet er der en tredje yderligere fil med udvidelsen ".**txt**".

- En fil med udvidelse **".h" ("header"),** der indeholder listen over alle tilgængelige funktioner, kommandoer og biblioteksdefinitioner;
- En fil med filtypenavnet **".cpp" ("c ++ kilde")** med den faktiske kode for de funktioner, der er præsenteret i oversigtsfilen.
- Én **keywords.txt-fil**, der gør det muligt for Arduino IDE at identificere bibliotekets funktioner og præsentere dem med en anden farve end resten af koden.

BnrOneA-biblioteket blev oprettet til at behandle den PIC18F45K22-tilknyttede hardware og giver Arduino mulighed for at interagere med det gennem SPI-kommunikationsbussen. Arduino har adgang til al hardware og funktionaliteter, der er defineret i biblioteket og i PIC18F45K22-softwaren. BnrOneA-biblioteket og PIC18F45K22-softwaren blev oprettet "den ene for den anden", og enhver ændring på den ene af dem kræver justering af den anden.

Alle BnrOneA-biblioteksfunktioner er specificeret og forklaret i afsnit 2 i denne manual.

1.1.3 C Programmeringssprog

C-sproget blev udviklet i 1972 af Dennis Ritchie på Bell Labs i New Jersey. Det opstod med ideen om at være et magtfuldt og hurtigt sprog, der skal bruges i UNIX-operativsystemet. Med tiden er det blevet forbedret og opdateret, hvilket gør det meget robust og pålideligt, og det begyndte at blive brugt af andre operativsystemer som Windows, MacOS og Linux. Det har været i konstant udvikling, siden det opstod i den første version kendt som "**K & R C**". I 1989 opstod den første specifikation som en standard fra det amerikanske institut for mønstre "**ANSI C**". I 1990 opstod "ISO C" af den internationale organisation for standardisering. I 1999 opstod standarden "**C99**", og den seneste revisionsdato i december 2011 til ISO / **IEC 9899: 2011**, bedre kendt som "C11".

Alle disse opdateringer og revisioner er rettet mod brugen af C-sproget til at udvikle programmer til personlige computere, hvor hukommelse og behandlingsressourcer ikke er en begrænsning. Til brug i mikrokontrollere bruges der en "lettere" version af C-sproget, fordi hukommelses- og behandlings-ressourcer er begrænsede. For at kunne programmere din Bot'n Roll ONE A skal du således kende et par grundlæggende regler for sproget til Arduino og hvordan nogle kommandoer kører.

Alle programmer til Arduino har to rutiner eller funktioner, som er obligatoriske. Konfigurationsrutinen "**setup** ()" udføres kun en gang ved starten af dit program. Her skal du skrive al koden til at initialisere variabler, indstille input og output pin, indstille SPI-kommunikation, Serial, I2C, faktisk alle de nødvendige konfigurationer.

Efter konfiguration kommer dit program ind i rutinen **loop** () og forbliver der på ubestemt tid. Udtrykket loop betyder cyklus, og i dette tilfælde er det en uendelig løkke, når programmet når slutningen af cyklussen går den tilbage til begyndelsen og starter igen! Det er her du skriver programmet!

C Programmering er ikke forklaret i denne vejledning, se eksemplerne på BnrOneA-biblioteket og Arduino generelt. Al koden er korrekt kommenteret, og du skal selv prøve og teste for at forstå, hvordan den fungerer. Vi giver dig dog nogle tip, vi har lært over tid på **botnroll.com**:

- Opret nye programmer ud fra de grundlæggende eksempler. Prøv at følge 3 eller 4 funktioner i robotten i det samme program fra de grundlæggende eksempler!
- Et program fungerer sjældent første gang! Tab ikke modet, analyser problemet og løs det!
- Indsæt kode systematisk, og test den, mens du følger med for at kontrollere, om alt kører som forventet.
- For at få et godt program, har du brug for mere tid til at test det, end for at skrive det!
- Brug værktøjer såsom debug-LED, seriel skærm eller LCD til at udskrive værdien af variabler og kontrollere, om programmet passerer et bestemt punkt i koden.
- Programmering er som at udøve en ny sport. I starten er det smertefuldt, fordi du ikke har den nødvendige fysiske tilstand, ikke kender reglerne, og det tager tid at forstå det. Ved at træne og øve, bliver du bedre og bedre. Ved ihærdigt arbejde, vil du efterhånden blive en ørn til det!

Bot'n Roll ONE A accepterer interaktion med en meget bred vifte af hardware. Der er ekstramateriale, ofte kaldet "skjolde" for Arduino, der gør næsten alt hvad du forestiller dig, og disse er kompatible med Bot'n Roll ONE A! Alle skjolde har biblioteker, der hjælper dig med at bruge og integrere, og kun din fantasi sætter grænsen!

2. BNRONEA ARDUINO LIBRARY-FUNKTIONER

BnrOneA-biblioteket har funktioner eller rutiner, der giver Arduino (ATmega328) adgang til alle de perifere enheder, der kontrolleres af PIC18F45K22. Disse rutiner er opdelt i tre grupper: konfiguration, læsning og skrivning. Mange af disse funktioner bruger **parametre/argumenter** til at udveksle information, dvs. sende og/eller modtage variable værdier.

Et argument er ethvert udtryk inden i parenteserne for en funktion, f.eks.

- **one.spiConnect (sspin);**

Argumentet er **sspin** og overføres som en **parameter** under udførelsen af **spiConnect**-funktionen.

En funktion kan også returnere en værdi som resultat af dens udførelse:

- **floatbatteri = one.readBattery ();**

Batterilæsefunktionen returnerer batteriets spænding som et resultat af funktionens udførelse. Batteriets værdi gemmes i variablen **batteri**.

I det følgende kan du læse listen over alle funktioner i BnrOneA-biblioteket, der er uddraget fra filen **BnrOneA.h**.

BnrOneA library Funktionsliste	
Setup rutiner	**Læserutiner**
void spiConnect(byte sspin);	byte obstacleSensors();
void minBat(float batmin);	byte readIRSensors();
void obstacleEmitters(boolean state);	IntreadAdc(byte);
	int readAdc0();
	int readAdc1();
	int readAdc2();
Skriverutiner	int readAdc3();
void servo1(byte position);	int readAdc4();
void servo2(byte position);	int readAdc5();
void led(boolean state);	int readAdc6();
void move(intspeedL,intspeedR);	int readAdc7();
void stop();	intreadButton();
void brake(byte torqueL,bytetorqueR);	float readBattery();
void resetEncL();	intreadEncL();
void resetEncR();	intreadEncR();
	intreadEncLInc();
	intreadEncRInc();
LCD Line 1 skriverutiner	**LCD Line 2 skriverutiner**
void lcd1(byte string[]);	void lcd2(byte string[]);
void lcd1(const char string[]);	void lcd2(const char string[]);
void lcd1(int number);	void lcd2(int number);
void lcd1(unsigned int number);	void lcd2(unsigned int number);
void lcd1(long int number);	void lcd2(long int number);
void lcd1(double number);	void lcd2(double number);
void lcd1(const char string[],int number);	void lcd2(const char string[],int number);
void lcd1(const char string[],unsigned int number);	void lcd2(const char string[],unsigned int number);
void lcd1(const char string[],long int number);	void lcd2(const char string[],long int number);
void lcd1(const char string[],double number);	void lcd2(const char string[],double number);
void lcd1(int num1, int num2);	void lcd2(int num1, int num2);
void lcd1(unsigned int num1, unsigned int num2);	void lcd2(unsigned int num1, unsigned int num2);
void lcd1(int num1, int num2, int num3);	void lcd2(int num1, int num2, int num3);
void lcd1(int num1, int num2, int num3, int num4);	void lcd2(int num1, int num2, int num3, int num4);
void lcd1(unsigned int num1, unsigned int num2, unsigned int num3);	void lcd2(unsigned int num1, unsigned int num2, unsigned int num3);
void lcd1(unsigned int num1, unsigned int num2, unsigned int num3, unsigned int num4);	void lcd2(unsigned int num1, unsigned int num2, unsigned int num3, unsigned int num4);

2.1.1 spiConnect(sspin)

Beskrivelse:

Initialiserer SPI-kommunikationsbussen ved at indstille SCK-, MOSI- og SS-stifter som output. Anbringer SCK- og MOSI-stifterne i lav tilstand (0V) og SS i høj tilstand (5V). I Bot'n Roll ONE A svarer SS-stiften "Slave Select" til SPI-kommunikation mellem ATmega328 og PIC18F45K22 som standard til den **digitale udgang 2**, men kan ændres ved at fjerne jumper SSP og oprette den ønskede forbindelse. Den indbyggede ATmega328 SS-pin kan bruges til SPI-kommunikation med ethvert skjold, der bruger dette link.

Parametere:

sspin: den digitale udgang, der skal bruges som "Slave Select" på SPI-kommunikationen mellem ATmega328 og PIC18F45K22. (Byte)

Returnerer:

Intet

Eksempel:

```
#include <BnrOneA.h>          // Bot'n Roll ONE A library
#include <SPI.h>              // SPI communication library, nødvendig til BnrOne.cpp
BnrOneA one;                  // deklarering ad objektvariabel til at kontrollere Bot'n Roll ONE A
#define SSPIN 2               // Slave Select (SS) pin til SPI-kommunikation
void setup()
{
    one.spiConnect(SSPIN);    // starter SPI kommunikationsmodul
}
```

2.1.2 minBat(batmin)

Beskrivelse

Definerer minimumværdien af batterispænding, hvor robotbevægelsen blokerer, og skriver en advarsel på anden linje på LCD-skærmen. Denne indstillingsværdi er skrevet til EEPROM på PIC18F45K22 og forsvinder ikke, når du lukker Bot'n Roll ONE A. Under skrivningen på EEPROM er PIC18F45K22 ikke tilgængelig, derfor skal der ikke sendes nogen kommandoer under processen, der varer cirka 5ms. Når batterispændingen kommer under minimumsindstillingen, reagerer robotten ikke på bevægelseskommandoer og viser en meddelelse om lavt batteriniveau på linje 2 på LCD-skærmen. Denne funktion hjælper med at bevare batteriets levetid og er meget nyttig i tilfælde af brug af lithium-batterier, der ødelægges, hvis spændingen falder til sikkerheds-minimumsværdien.

Parametre:

batmin: Minimumspændingen for batteriets funktion(float)

Returnerer:

Intet

Eksempel:

```
#include <BnrOneA.h>          // Bot'n Roll ONE A library
#include <SPI.h>              // SPI kommunikations-library nødvendig til BnrOne.cpp
BnrOneA one;                  // deklarering af objektvariablen til kontrol af Bot'n Roll ONE A
#define SSPIN 2               // Slave Select (SS) pin til SPI-kommunikation
void setup()
{
    one.spiConnect(SSPIN);    // starter the SPI kommunikationsmodul
    one.minBat(8.5);          // definerer mindste batterispænding
    delay(5);                         // venter 5ms
}
```

2.1.3 obstacleEmitters(state)

Beskrivelse:

Kontrollerer tilstanden for de infrarøde LED-emittere. De infrarøde emittere kan slukkes om nødvendigt, og normal registrering af forhindringer er deaktiveret. Værdien 0 slukker emitterne, og værdien 1 aktiverer emitterne.

Parametre:

state: tilstanden, der skal placers på LED'erne (*boolean*)

Returerer:

Intet

Example:

```
#include <BnrOneA.h>              // Bot'n Roll ONE A library
#include <SPI.h>                  // SPI kommunikations-library nødvendig for BnrOne.cpp
BnrOneA one;                      // deklaration af objektvariabel, til kontrol af Bot'n Roll ONE A
#define SSPIN 2                   // Slave Select (SS) pin for SPI-kommunikation
void setup()
{
    one.spiConnect(SSPIN);        // starter SPI kommunikationsmodukl
    one.obstacleEmitters(ON);     // aktiverer IR emitter LEDs
}
```

2.2 LÆSEFUNKTIONER

2.2.1 obstacleSensors()

Beskrivelse:

Returnerer resultatet læst fra forhindringsf'lerne, og der er fire mulige situationer:

- **0**: der er ingen forhindringer
- **1**: forhindring registreret på venstre sensor
- **2**: forhindring registreret på højre sensor
- **3**: forhindringer registreret på både venstre og højre sensor

Forhindringslæsning udføres hver 20ms af PIC18F45K22, og denne funktion returnerer resultatet af den sidste aflæsning. For at muliggøre læsning af forhindringer er det nødvendigt, at de infrarøde emittere aktiveres!

Parametre:

Ingen

Returnerer:

Den læste værdi af forhindringsfølerne (*byte*)

Eksempler:

```
...
void loop()
{
    byte obstacle=one.obstacleSensors(); // læser forhindringsfølerne
...
}
```


2.2.2 readIRSensors()

Beskrivelse:

Returnerer den aktuelle tilstand af SHARP GP1UX511QS infrarøde sensorer, og der er fire mulige situationer:

- 0: begge sensorer på "Høj"
- 1: venstre sensor "Høj" og højre sensor "Lav"
- 2: venstre sensor "Lav" og højre sensor "Høj"
- 3: begge sensorer "Lav"

Denne funktion tvinger PIC18F45K22 til at foretage en øjeblikkelig aflæsning af de infrarøde sensors tilstand. Bemærk, at SHARP GP1UX511QS-sensorerne har et inverteret signal i forhold til den infrarøde sender.

Parametre:

Ingen

Returnerer:

Den aktuelle tilstand på de infrarøde sensorer (*byte)*

Eksempel:

```
...
void loop()
{
    byte obstacle=one.readIRSensors(); // læser aktuel status på IR-sensorerne
...
}
```


2.2.3 readAdc(byte)

Beskrivelse:

Returnerer værdien af ADC-konvertering "Analog til digital konvertering" tilknyttet PIC18F45K22 for den funktion, der er specificeret i parameterkanalen. Parameteren definerer, til hvilke af de analoge kanaler AN0 til AN7, Bot'n Roll ONE A, du ønsker konverteringen, og accepterer følgende værdier:

- 0 (svarer til kanal AN0)
- 1 (svarer til kanal AN1)
- 2 (svarer til kanal AN2)
- 3 (svarer til kanal AN3)
- 4 (svarer til kanal AN4)
- 5 (svarer til kanal AN5)
- 6 (svarer til kanal AN6)
- 7 (svarer til kanal AN7)

Den opnåede værdi ved konverteringen varierer mellem 0 og 1023 for PIC18F45K22, da den har en ADB-konverter på 10 bit.

Parametere:

ADC-kanalen (*byte*)

Returnerer:

Værdien fra ADC konverteringen (int)

Eksempel:

```
...
void loop()
{
    for(i=0;i<8;i++)
    {
    adc[i]=one.readAdc(i);        // læser ADC-konverteringsværdien
    ...
    }
}
```


2.2.4 readAdcX()

Beskrivelse:

Returnerer værdien af ADC-konvertering "Analog til digital konvertering" tilknyttet PIC18F45K22 fra ANX-kanalen i Bot'n Roll ONE A. Værdien opnået i konverteringen varierer mellem 0 og 1023 for PIC18F45K22, da den har 10 bit ADC-konverter.

"X" -bogstavet svarer til ADC-læsekanalen, der er ønskes læst og skal erstattes af et tal fra 0 til 7.

Parametre:

Ingen

Returnerer:

Værdien af ADC-konverteringen (int)

Eksempel:

```
...
void loop()
{
    int adc1 =one.readAdc1();   // læser ADC-konverteringsværdien på kanal 1
    int adc6 =one.readAdc6();   // læser ADC-konverteringsværdien på kanal 6
...
}
```


2.2.5 readButton()

Beskrivelse:

Angiver, hvilken af trykknapperne PB1, PB2 eller PB3, der trykkes på. Funktionen returnerer en af de mulige værdier som et resultat:

- 0: der trykkes ikke på nogen knap
- 1: PB1 er trykket
- 2: PB2 er trykket
- 3: PB3 er trykket

Hvis der trykkes på mere end en trykknap på samme tid, returneres den laveste værdi.

Parametre:

Ingen

Returnerer:

Tallet for den aktiverede knap (*int*)

Example:

```
...
void loop()
{
    int pbutton=one.readButton();     // læser trykknappens værdi
...
}
```

2.2.6 readBattery()

<beskrivelse:

Læser batteriets spænding I volt.

Parametre:

Ingen

Returnerer:

Batterispændingen i Volt (float)

Eksempel:

...


```
void loop()
{
    float battery=one.readBattery();          // læser batterispændingen
...
}
```

2.2.7 readEncL()

Beskrivelse:

Returnerer tællingen for den venstre koder, og koderen udfører en nulstilling, nulstiller tællerværdien. Denne funktion er nyttig til at måle og kontrollere hastigheden på et hjul. Hvis hjulets hastighed er konstant, bør vi i veldefinerede tidsintervaller opnå de samme antal værdier.

Koderen er på 16 bit, og tallet varierer mellem -32768 og 32767.

Parametre:

Ingen

Returnerer:

Tallet i den venstre koder (*int*)

Eksempel:

```
...
void loop()
{
    int encL=one.readEncL();
...
}
```

2.2.8 readEncR()

Beskrivelse:

Returnerer tællingen for den højre koder, og koderen udfører en nulstilling, nulstiller tællerværdien. Denne funktion er nyttig til at måle og kontrollere hastigheden på et hjul. Hvis hjulets hastighed er konstant, bør vi i veldefinerede tidsintervaller opnå de samme antal værdier.

Koderen er på 16 bit, og tallet varierer mellem -32768 og 32767.

Parametre:

Ingen

Returnerer:

Tallet i den højre koder (*int*)

```
{
    int encR=one.readEncR();
...
}
```


2.2.9 readEncLInc()

Beskrivelse:

Returnerer aflæsningen af den trinvise tælling for den venstre koder. Denne funktion er nyttig til at styre den strækning, et hjul kører. Når man ved, at øgningen med en enhed af værdien af koderen svarer til en bestemt afstand, kan man kontrollere afstanden, der er kørt af et hjul. Der udføres ingen nulstilling til koderværdien. Koderen er en 16-bit tæller, og antallet er mellem -32768 og 32767. Vær opmærksom på, at der ikke kommer overløb.

Parametere:

Ingen

Returnerer:

Stigende værdi af den venstre koder (*int*)

Eksempel:

```
…
void loop()
{
    int encL=one.readEncLInc();
…
}
```

2.2.10 readEncRInc()

Beskrivelse:

Returnerer aflæsningen af den trinvise tælling for den højre koder. Denne funktion er nyttig til at styre den strækning, et hjul kører. Når man ved, at øgningen med en enhed af værdien af koderen svarer til en bestemt afstand, kan man kontrollere afstanden, der er kørt af et hjul. Der udføres ingen nulstilling til koderværdien. Koderen er en 16-bit tæller, og antallet er mellem -32768 og 32767. Vær opmærksom på, at der ikke kommer overløb.

Parametere:

Ingen

Returnerer:

Stigende værdi af den højre koder (*int*)

Eksempel:

```
…
void loop()
{
    int encR=one.readEncRInc();
…
}
```


2.3 KOMMANDOFUNKTIONER

2.3.1 servo1(position)

Beskrivelse:

Flytter servoen, der er tilsluttet SER1. Funktionsparameteren definerer vinkelpositionen, der varierer fra 0 til 180.

En standard servo positionerer sig selv i en defineret vinkel, der sendes gennem en funktionsparameter, og vinklen varierer mellem 0̊ og 180̊.

En kontinuerlig rotationsservo varierer dens rotationshastighed i henhold til den værdi, der sendes gennem funktionsparameteren. '0' svarer til den maksimale rotation i en bestemt retning, 180 svarer til det maksimale i den modsatte retning og 90 svarer til at servoen stoppes.

Parametre:

position: Servoens vinkelposition (*byte*)

Returnerer:

Intet

Eksempel:

```
...
void loop()
{
    one.servo1(70);
...
}
```


2.3.2 servo2(position)

Beskrivelse:

Flytter servoen, der er tilsluttet SER2. Funktionsparameteren definerer vinkelpositionen, der varierer fra 0 til 180.

En standard servo positionerer sig selv i en defineret vinkel, der sendes gennem en funktionsparameter, og vinklen varierer mellem 0̊ og 180̊.

En kontinuerlig rotationsservo varierer dens rotationshastighed i henhold til den værdi, der sendes gennem funktionsparameteren. '0' svarer til den maksimale rotation i en bestemt retning, 180 svarer til det maksimale i den modsatte retning og 90 svarer til at servoen stoppes.

Parametre:

position: Servoens vinkelposition (*byte*)

Returnerer:

Intet

Eksempel:

```
...
void loop()
{
    one.servo2(130);
...
}
```


2.3.3 led(state)

Beskrivelse:

Tænder eller slukker for Bot'n Roll ONE A LED. LED-status er defineret af den parameter, der sendes til funktionen og har to mulige tilstande:

• 0: LED slukket

• 1: LED tændt

Parametre:

state: LED'ens status (*boolean*)

Returnerer:

Intet

Eksempel:

```
…
void loop()
{
    one.led(HIGH);          // sætter LED ON
…
}
```

2.3.4 move(speedL,speedR)

Beskrivelse:

Bevæger Bot'n Roll ONE A-motorerne. Parametrene definerer hastigheden for hver motor, venstre og højre, der spænder fra -100 til 100. Værdien -100 svarer til den maksimale omdrejningstal i baglæns, 100 svarer til den maksimale hastighed i fremadretningen og 0 stopper motorerne.

Parameters:

speedLvenstre motors hastighed (*int*)

speedR: Højre motors hastighed (int)

Returnerer:

Intet

Eksempel:

```
…
void loop()
{
    one.move (70,70);          // Bevæger begge motorer fremad med hastighed 70.
…
```



```
}
```

2.3.5 stop()

Description:

Det stopper begge motorer på Bot'n Roll ONE A. Det lukker for energien til motorerne, men de roterer frit, indtil de stopper, uden at de bremses.

Parametere:

Ingen

Returnerer:

Intet

Eksempel:

```
...
void loop()
{
    one.stop ();          // Stopper begge motorer uden bremsning
...
}
```

2.3.6 brake(torqueL,torqueR)

Beskrivelse:

Stop Bot'n Roll ONE A-motorerne, og bremser. Bremsekraften for hver motor er defineret af funktionsparametrene og varierer mellem 0 og 100. Værdien 0 svarer til stop uden bremsemoment. Værdien 100 svarer til stop med maksimalt bremsemoment og motorblokkene øjeblikkeligt!

Parameters:

torqueL: venstre motors bremseeffekt (*byte*)

torqueR: højre motors bremseeffekt (*byte*)

Returnerer:

Intet

Eksempel:

```
...
void loop()
{
    one.brake (60,60);          // Bremser begge motorer med 60% bremsekraft.
...
}
```


2.3.7 resetEncL()

Beskrivelse:

Nulstiller den venstre koder ved at sætte dens værdi til 0.

Parametere:

Ingen

Returnerer:

Intet

Eksempel:

```
...
void loop()
{
    one.resetEncL();        // Reset venstre koder
...
}
```

2.3.8 resetEncR()

Beskrivelse:

Nulstiller den højre koder ved at sætte dens værdi til 0.

Parametere:

Ingen

Returnerer:

Intet

Eksempel:

```
...
void loop()
{
    one.resetEncR();        // Reset højre koder
...
}
```


2.4 LCD SKRIVEFUNKTIONER

2.4.1 lcdX(string[])

Beskrivelse:

Udskriver på X-linjen på LCD-skærmen en matrix med tegn (streng) eller tekst i anførselstegn ("tekst til at sende") sendt som parameter. Parametre accepteres som variabeltypen (char) og (constchar). Maksimal størrelse i tegn for en streng eller tekst i anførselstegn er 17, der svarer til 16 skrevne tegn på LCD-skærmen plus det afsluttende tegn '\ 0'.

"X" -bogstavet repræsenterer LCD-linjen, som brugeren ønsker at skrive på, og skal være 1 eller 2.

Parametre:

string[]: array med de karakterer, der skal skrives på LCD'en (*char*) or (*constchar*).

Returnerer:

Intet

Eksempel:

```
...
char string[]=" String Test ";       // declarer og initaliserer strengen

{
     one.lcd1(string);                //skriver strengen på LCD linje 1
     one.lcd2(" Text to LCD! ");          //skriver strengen på LCD lijne 2
...
}
```


2.4.2 lcdX(number)

Beskrivelse:

Udskriver på X-linjen på LCD-skærmen et nummer eller en variabel, der sendes som en parameter. Nummeret kan være en variabel type (int) (unsigned int) (long int) (double) eller (float).

"X" -bogstavet repræsenterer LCD-linjen, som brugeren agter at skrive på, og skal være 1 eller 2.

Parametre:

number: tal eller variable der skal skrives (*int*), (*unsigned int*), (*long int*), (*double*) eller (*float*).

Returnerer:

Intet

Eksempel:

```
...
intvar1 =-32768;                // Deklarerer og initaliserer variablen
unsigned int var2 = 0;          // Deklarerer og initaliserer variablen
long int var3 = - 2147483648;   // Deklarerer og initaliserer variablen
float var4=123.12;              // Deklarerer og initaliserer variablen
double var5=123.12;             // Skriver variablens værdi på LCD'en

void loop()
{
    one.lcd1(var1);             //Skriver variablens værdi på LCD'en
    ...
    one.lcd2(var2);             // Skriver variablens værdi på LCD'en
    ...
    one.lcd1(var3);             // Skriver variablens værdi på LCD'en
    ...
    one.lcd2(var4);             // Skriver variablens værdi på LCD'en
    ...
    one.lcd1(var5);             // Skriver variablens værdi på LCD'en
    ...
    one.lcd2(32767);            // Skriver variablens værdi på LCD'en
    ...
    one.lcd1(2147483647);       // Skriver variablens værdi på LCD'en
    ...
    one.lcd2(321.01);                   //Udskriver enkelt tal på LCD'en
...
}
```


2.4.3 lcdX(string[],number)

Beskrivelse:

Udskriver på X-linjen på LCD-skærmen den tekst og det nummer, der er sendt som parametre. Som tekst accepteres tekstvariabel (constchar) eller tekst mellem anførselstegn. Det samlede antal tegn, der skal udskrives, må ikke være større end 16. Antallet kan være en variabel type (int) (unsigned int) (long int) (double) eller (float).

"X" -bogstavet repræsenterer LCD-linjen, som brugeren agter at skrive, og skal være 1 eller 2.

Parametre:

string[]: Karakter, der skal skrives på LCD'en (*constchar*).

number: tal eller variabel, der skal skrives (*int*), (*unsigned int*), (*long int*), (*double*) eller (*float*).

Returnerer:

INtet

Eksempel:

```
...
intvar1 =-32768;                    // Deklarerer og initaliserer variablen
unsigned int var2 = 0;              // Deklarerer og initaliserer variablen
long int var3 = - 2147483648;       // Deklarerer og initaliserer variablen
float var4=123.12;                  // Deklarerer og initaliserer variablen
double var5=123.12;                 // Deklarerer og initaliserer variablen

void loop()
{
    one.lcd1("Text: ", var1);               // Skriver tekst og variabelværdien på LCD
    ...
    one.lcd2("Text: ", var2);               // Skriver tekst og variabelværdien på LCD
    ...
    one.lcd1("Text: ",var3);                // Skriver tekst og variabelværdien på LCD
    ...
    one.lcd2("Text: ",var4);                // Skriver tekst og variabelværdien på LCD
    ...
    one.lcd1("Text: ",var5);                // Skriver tekst og variabelværdien på LCD
    ...
    one.lcd2("Text: ",32767);               // Skriver tekst og variabelværdien på LCD
    ...
    one.lcd1("Text: ",2147483647);          // Skriver tekst og variabelværdien på LCD
    ...
    one.lcd2("Text: ",321.01);              //Skriver tekst og et enkelt tal på LCD
...
}
```


2.4.4 lcdX(num1, num2)

Beskrivelse:

Udskrives på X-linjen på LCD tal eller variabler sendt som parametre. Tal kan være variabler af type (int) eller (unsigned int)."

X" -bogstavet repræsenterer LCD-linjen, som brugeren agter at skrive på, og skal være 1 eller 2.

Parametre:

num1: tal eller variabel, der skal skrives (*int*) eller (*unsigned int*).

num2: tal eller variabel, der skal skrives (*int*) eller (*unsigned int*).

Returnerer:

Intet

Eksempel:

```
...
intvar1 =-32768;                // deklarerer og initaliserer variablen
unsigned int var2 = 0;          // deklarerer og initaliserer variablen

void loop()
{
    one.lcd1(var1 , 32767);             //skriver variabel og tal på LCD
    ...
    one.lcd2(var2, 65535);              // skriver variabel og tal på LCD
...
}
```


2.4.5 lcdX(num1, num2, num3)

Beskrivelse:

Udskrives på X-linjen på LCD tal eller variabler sendt som parametre. Tal kan være variabler af type (int) eller (unsigned int).

"X" -brevet repræsenterer LCD-linjen, som brugeren ønsker at skrive på, og skal være 1 eller 2.

Parametre:

num1: tal eller variable der skal skrives (*int*) eller (*unsigned int*).

num2: tal eller variable der skal skrives (*int*) eller (*unsigned int*).

num3: tal eller variable der skal skrives (*int*) eller (*unsigned int*).

Returnerer:

Intet

Eksempel:

```
...
intvar1=-32768;                 // deklarerer og initaliserer variabelen
unsigned int var2 = 0;          // deklarerer og initaliserer variabelen

void loop()
{
        one.lcd1(var1 ,32767, var2);          //skriver variabelens værdi og tallet på LCD
...
}
```


2.4.6 lcdX(num1 , num2 , num3 , num4)

Beskrivelse:

Udskrives på X-linjen på LCD tal eller variabler sendt som parametre. Tal kan være variabler af type (int) eller (unsigned int).

"X" -brevet repræsenterer LCD-linjen, som brugeren ønsker at skrive på, og skal være 1 eller 2.

Parametre:

num1: tal eller variabel, der skal skrives (*int*) eller (*unsigned int*).

num2: tal eller variabel, der skal skrives (*int*) eller (*unsigned int*).

num3: tal eller variabel, der skal skrives (*int*) eller (*unsigned int*).

num4: tal eller variabel, der skal skrives (*int*) eller (*unsigned int*).

Returnerer:

Intet

Eksempel:

```
...
intvar1 =-32768;                    // deklarerer og initaliserer variabelen
unsigned int var2 = 0;              // deklarerer og initaliserer variabelen

void loop()
{
    one.lcd2(var1 ,32767, var2 , 65535);    //Skriver variabelens værdi og tallet på LCD
...
}
```


ANNEX A: INSTALLERING AF USB-SERIEL (RS232) KONVERTER VCP-DRIVER

Driveren tillader din computers operativsystem at kommunikere med Bot'n Roll ONE A.

For at installere driveren skal du besøge Bot'n Roll ONE A supportwebsiden http://botnroll.com/onea/ og downloade "**VCP Driver - Windows" eller "VCP Driver - Mac OS X**" ved at klikke på dt rigtige link i forhold til til dit operativsystem. Efter download skal du dekomprimere ".zip" -filen og køre applikationen.

Hver gang du slutter din robot til din computer ved hjælp af USB-kablet oprettes en virtuel COM-port (VCP), gennem hvilken kommunikationen mellem din Bot'n Roll ONE A og din pc udføres. Programmeringsapplikationsmiljøet bruger porten til at kommunikere med Bot'n Roll ONE A og overfører derfor dine programmer til din robot.

USB-serielkonverteren, der bruges af Bot'n Roll ONE A, er et **PoUSB12**-produkt fra PoLabs og bruger **CP2102 Bridge**-enheden fra Silicon Labs.

ANNEX B: ARDUINO IDE

Den *software*, der bruges til at programmere robotten, er Arduino IDE. Denne applikation er nødvendig for at redigere programmerne på C-sprog. Det bruges også til at overføre dine programmer til Bot'n Roll ONE A.

B.1 ARDUINO IDE INSTALLATION

For at kunne installere Arduino IDE skal du besøge Bot'n Roll ONE A-supportwebsiden http://botnroll.com/onea/ og klikke på det linket "**Arduino IDE - Windows**" eller "**Arduino IDE - Mac OS X**" for at downloade det i henhold til dit operativsystem.

Efter download skal du dekomprimere .zip-filen og kopiere den til en mappe efter eget valg på din computer.

Denne mappe indeholder flere undermapper og filer, blandt dem "**arduino**" -programmet, der starter Arduino IDE. Undermappen "***libaries***" er også meget vigtig og indeholder alle Arduino-biblioteker. Bibliotekerne er dine programmeringsværktøjer.

B.2 INSTALLERING THE **BNRONEA** LIBRARY PÅ ARDUINO

BnrOneA-biblioteket, der er udviklet af **botnroll.com** til Arduino IDE, indeholder alle de nødvendige kommandoer til at kontrollere robotten. Dette bibliotek skal installeres på Arduino IDE.

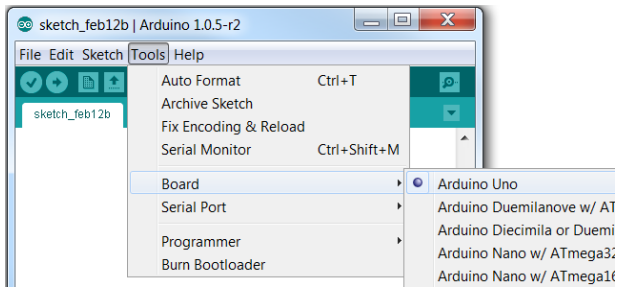
På Bot'n Roll ONE A supportwebsiden http://botnroll.com/onea/ skal du downloade filen **BnrOneA.zip** ved at klikke på "**Arduino Library**".

Dekompresser filen, og anbring den udpakkede mappe "**BnrOneA**" i biblioteket "***libaries***", der blev beskrevet i det foregående afsnit. Nu har du alle de nødvendige værktøjer til at programmere din **Bot'n Roll ONE A** med succes!

B.3 KONFISURERING AF KOMMUNIKATIONEN MED ROBOTTEN

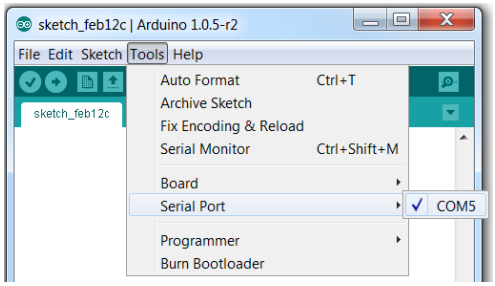
Før du fortsætter med dette trin, skal du sikre dig, at du har installeret VCP-driveren korrekt. Tilslut Bot'n Roll ONE A til din computer ved hjælp af det medfølgende USB-kabel. På dette tidspunkt tildeles automatisk en COM-port til kommunikation med robotten.

Åbn Arduino IDE, tryk på "**Værktøjer -> Board**", og vælg "**Arduino Uno**" -kortet. **Bot´n Roll ONE A** programmeres som om det var en Arduino Uno.



Figur 2: Valg af board til programmering

Under indstillingen "Værktøjer -> Port" skal du vælge den korrekte COM-port, der er givet til Bot'n Roll ONE A.



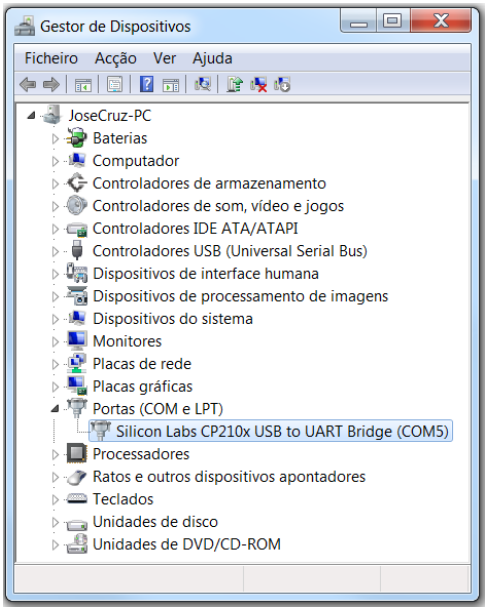
Figur 3: Valg af seriel port

Hvis der ikke er nogen COM-port tilgængelig, har du sandsynligvis ikke installeret USB-Serial converter VCP-driveren korrekt.

Åbn Windows-enhedhåndtering, og søg efter emnet "Porte (COM og LPT)". Udvid punktet, så ser du alle tilgængelige COM-porte.

"**Silicon Labs CP210x USB to UART Bridge**" er betegnelsen, der identificerer den COM-port, der "taler" med Bot'n Roll ONE A (på dette figureksempel er **COM5**-porten blevet tildelt).

Hvis punktet "**Silicon Labs CP210x USB til UART Bridge**" ikke vises på listen, skal du installere VCP-driveren korrekt.

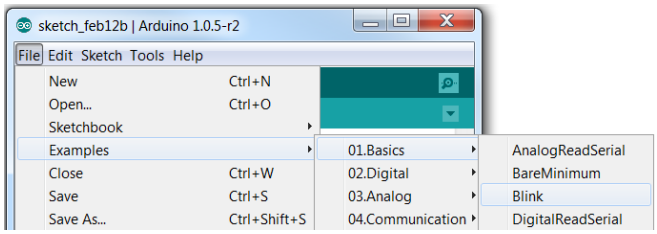


Figur 4: COM-porte i Enhedshåndtering

B.4 INDLÆSE ET PROGRAM TIL BOT'N ROLL ONE A

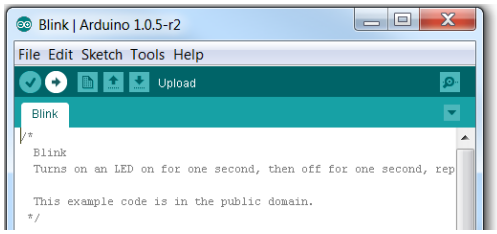
På Arduino IDE-applikationen finder du flere eksempler på programmer, som du kan indlæse til din robot.

Klik på "**File -> Eksempler -> 01.Basics -> Blink**", og et nyt vindue vises med koden til dette eksempel.



Figur 5: Indlæse et eksempelprogram

Klik på "**Sketch -> Upload**" eller tryk på ikonet "pil der peger til højre" for at sende programmet til robotten. Så snart uploaden er afsluttet, skal du se den gule L-LED blinke en gang i sekundet!



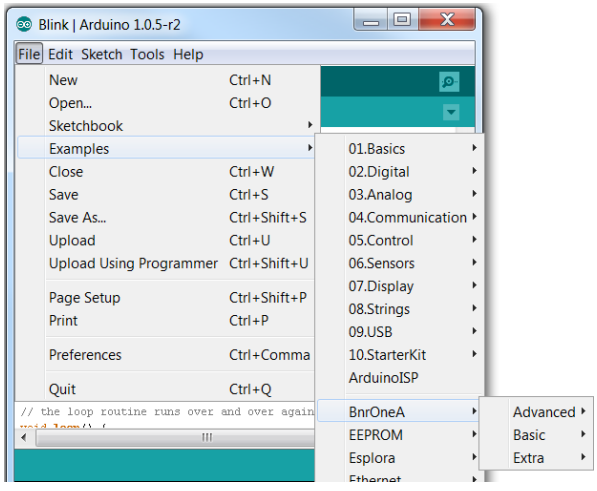
Figur 6: et program sendes til robotten

Hvis du klikker på "***File -> Eksempler -> BnrOneA ->*** ..." vil du finde alle programmer udviklet af botnroll.com specifikt til Bot'n Roll ONE A.

I "***Filer -> Eksempler -> BrOneA -> Basic -> ...***" kan du finde de enkleste programmer, hvis hovedformål er at teste al din robothardware. Du skal studere og forstå alle disse små programmer!

På "***Filer -> Eksempler -> BnrOneA -> Avanceret ->... ***" findes nogle mere avancerede programmer, som du muligvis først forstår efter de enkle.

I "***Filer -> Eksempler -> BnrOneA -> Extra -> ...***" kan du finde de programmer, der er relateret til Bot'n Roll ONE A-ekstra-materiale, der vil gøre **Bot´n Roll ONE A** til en mere kraftfuld robot.



Figur 7: Programmer fra BnrOneA librariet

