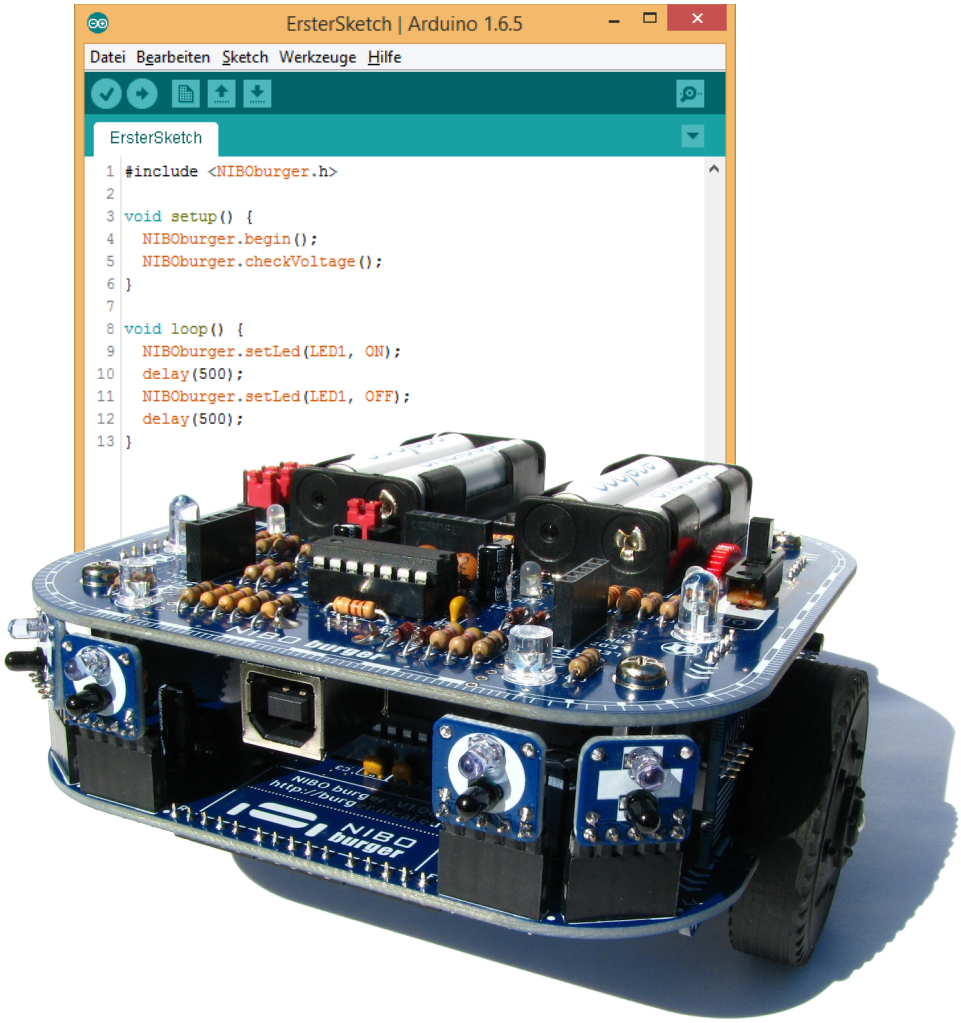
Robot Kit NIBO burger

ARDUINO 1.6x Programming-Tutorial



nicai systems

version 2015/09/09

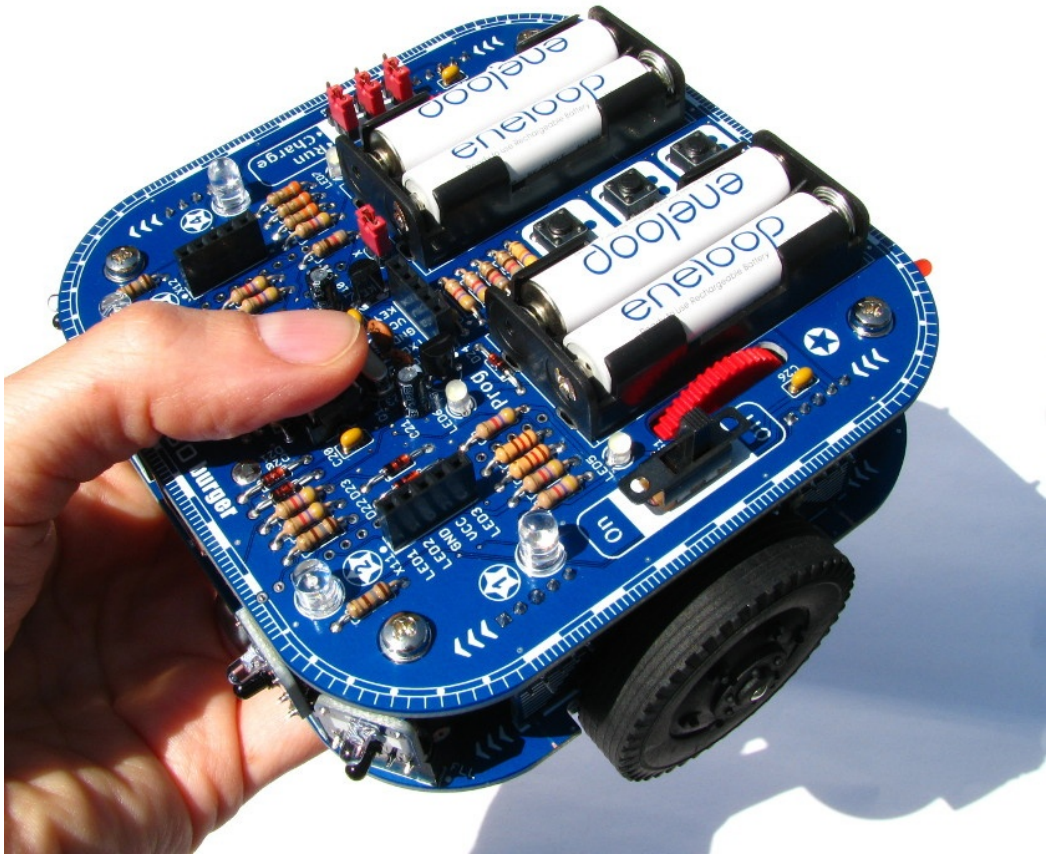
Table of contents

1 Introduction

This tutorial provides a step-by-step manual for first operations with the NIBO burger with **ARDUINO**. With small, detailed explained examples the user gets in touch with all basic functions of the robot.

In the following sections you learn how to install the development environment, how the leds can be enlightened, how the colour sensors, the IR-sensors and the motors can be controlled and how the NIBO burger is able to move!

The tutorial is made for beginners in the fields robotic and programming. You'll get a quick and motivating access to programming and especially to microcontroller-programming.



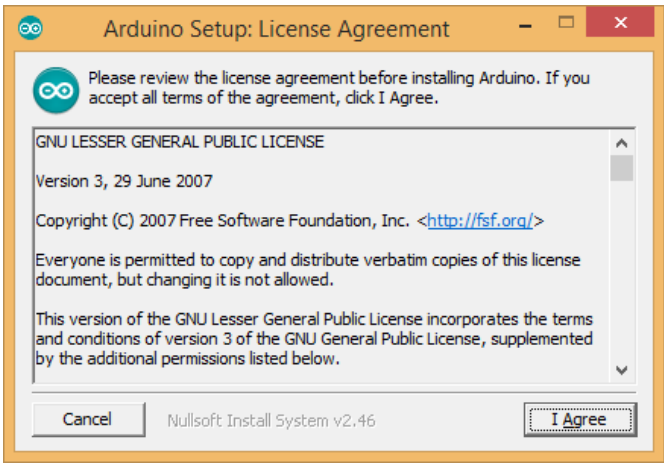
2 Installing the Development Environment

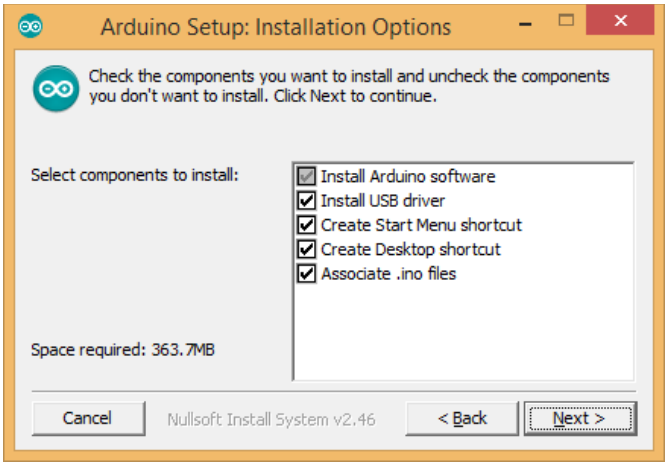
2.1 ARDUINO

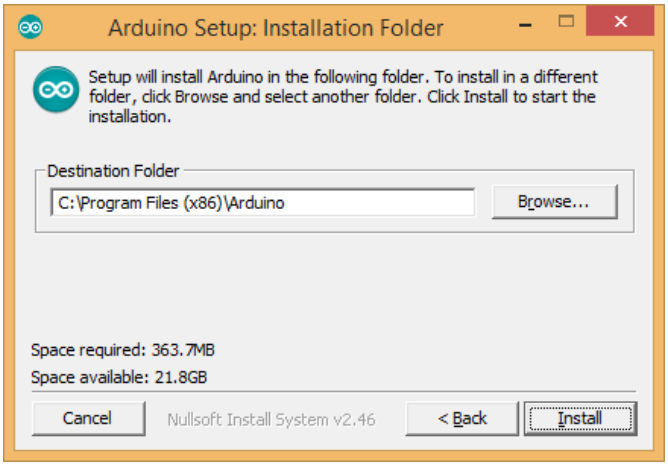
First of all you have to install the ARDUINO. Therefore the installation file *Arduino 1.6.x* must be downloaded from **http://arduino.cc**. You have to choose your individual operating system. For Windows you must choose the installer version.

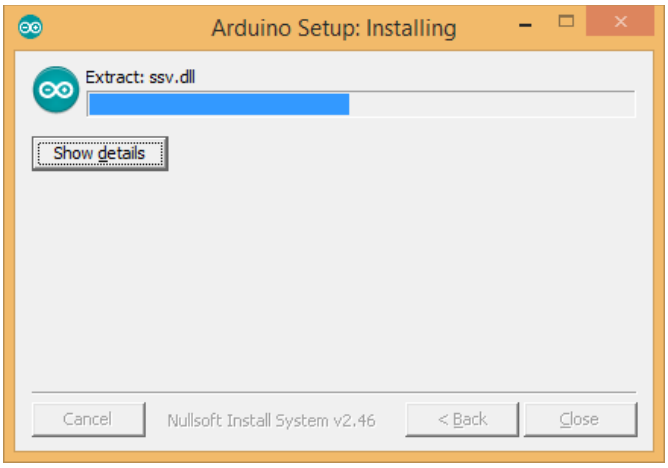
The following section describes the installation for Windows:

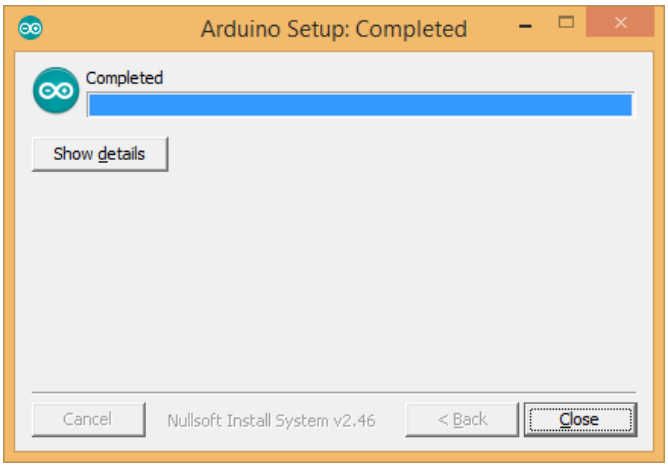
The installer can be started by double-click on the downloaded file ***arduino-1.6.x-windows.exe***. Now you can follow the instructions of the installer:






Arduino Setup: Installation Folder
Setup will install Arduino in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.
Destination Folder
C:\Program Files (x86)\Arduino
Browse...
Space required: 363.7MB
Space available: 21.8GB
Cancel
Nullsoft Install System v2.46
< Back
Install


Arduino Setup: Installing
Extract: ssv.dll
Show details
Cancel
Nullsoft Install System v2.46
< Back
Close


Arduino Setup: Completed
Completed
Show details
Cancel
Nullsoft Install System v2.46
< Back
Close

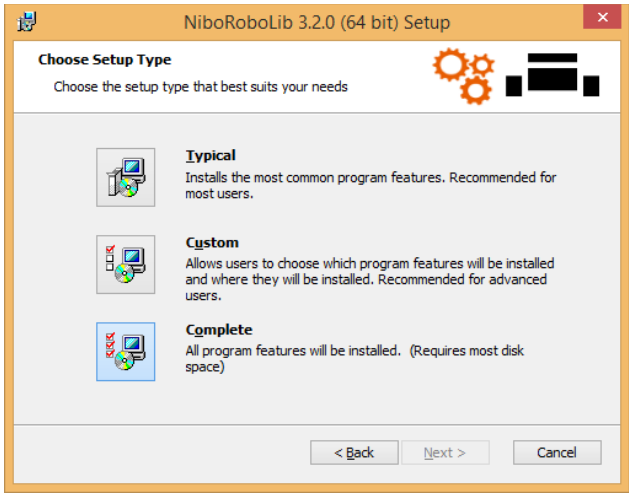
3 Installation of the NiboRoboLib

Now the NiboRoboLib has to be installed. The **latest** version and an **installation manual** (.pdf) are to find here:

http://www.roboter.cc/niboRoboLib

All files are also available on the enclosed CD.

The best method to get all needed packages is to do the complete installation:

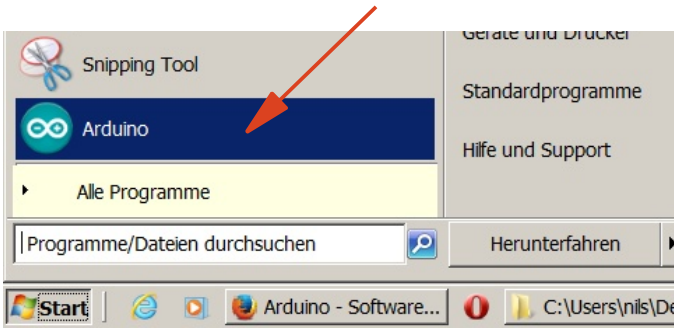


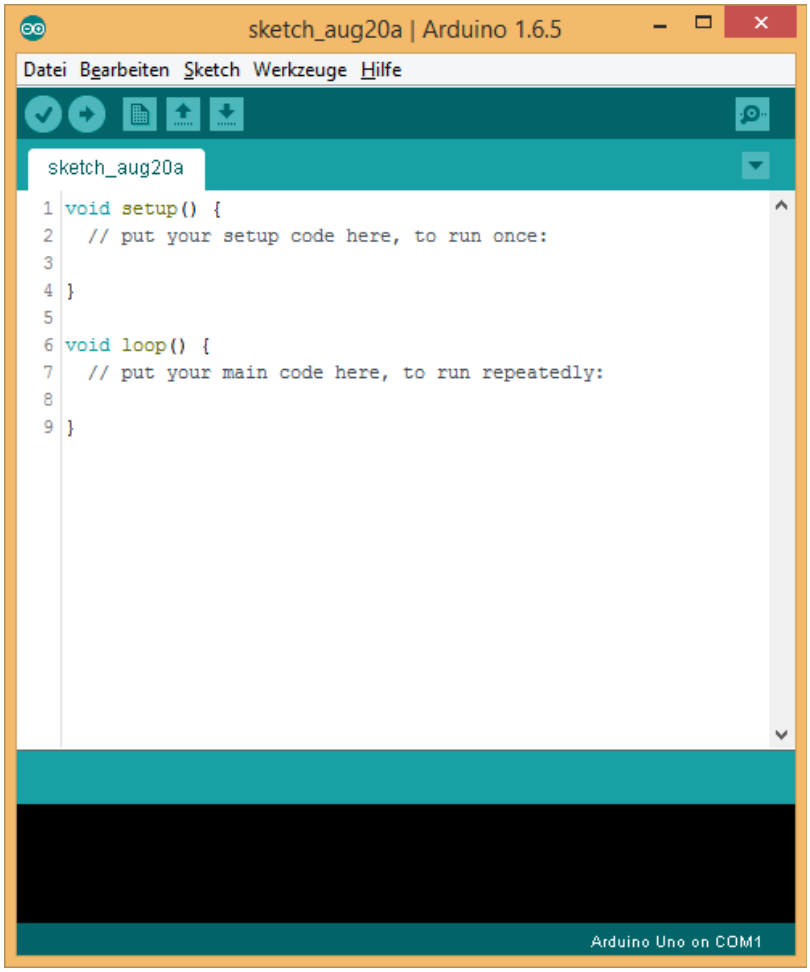
The **NiboRoboLib** contains:

- All necessary **drivers** for UCOM-IR2-X, NIBObee and NIBO burger
- **RoboDude** (transmission program for **.hex**- and **.xhex**-files)
- **C-library**, **test- and calibrating-programs** for NIBO 2, NIBO burger and NIBObee
- **ARDUINO**-library for NIBO burger, NIBObee and NIBO 2

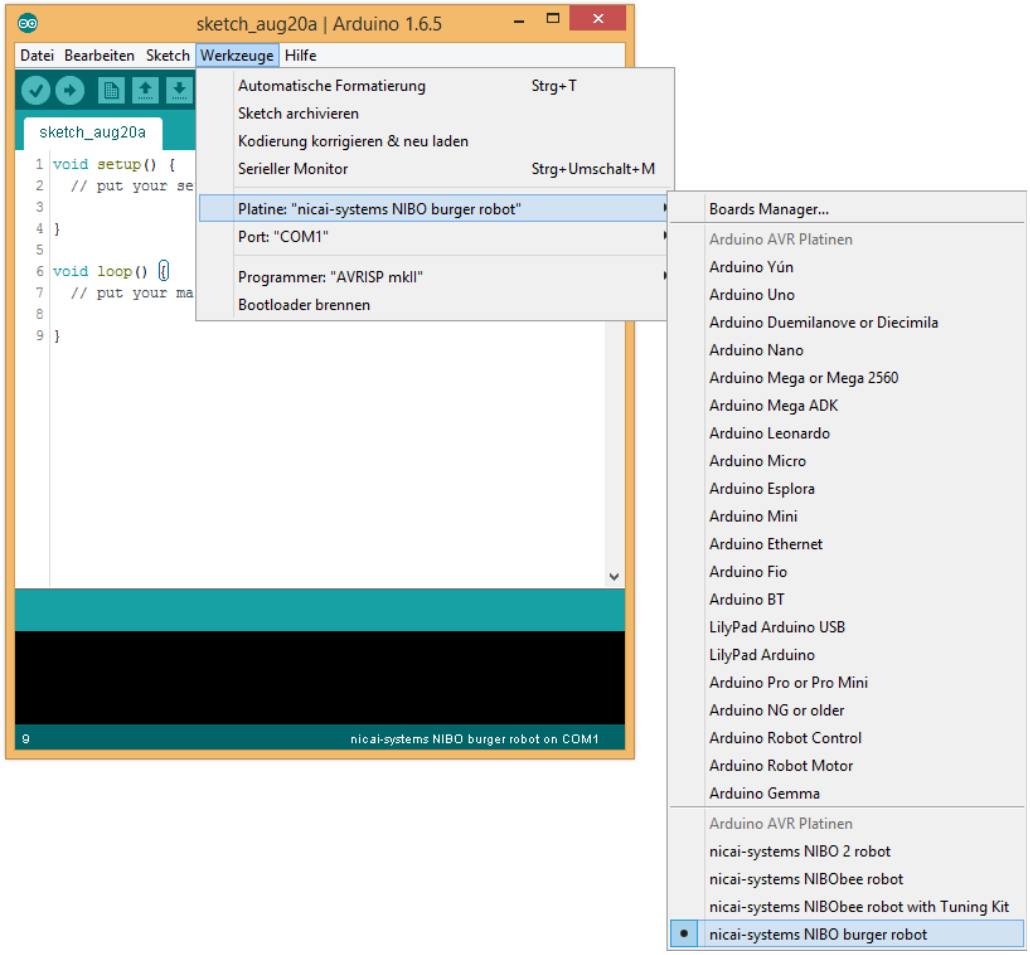
4 Preparations

Now we start the ARDUINO environment:

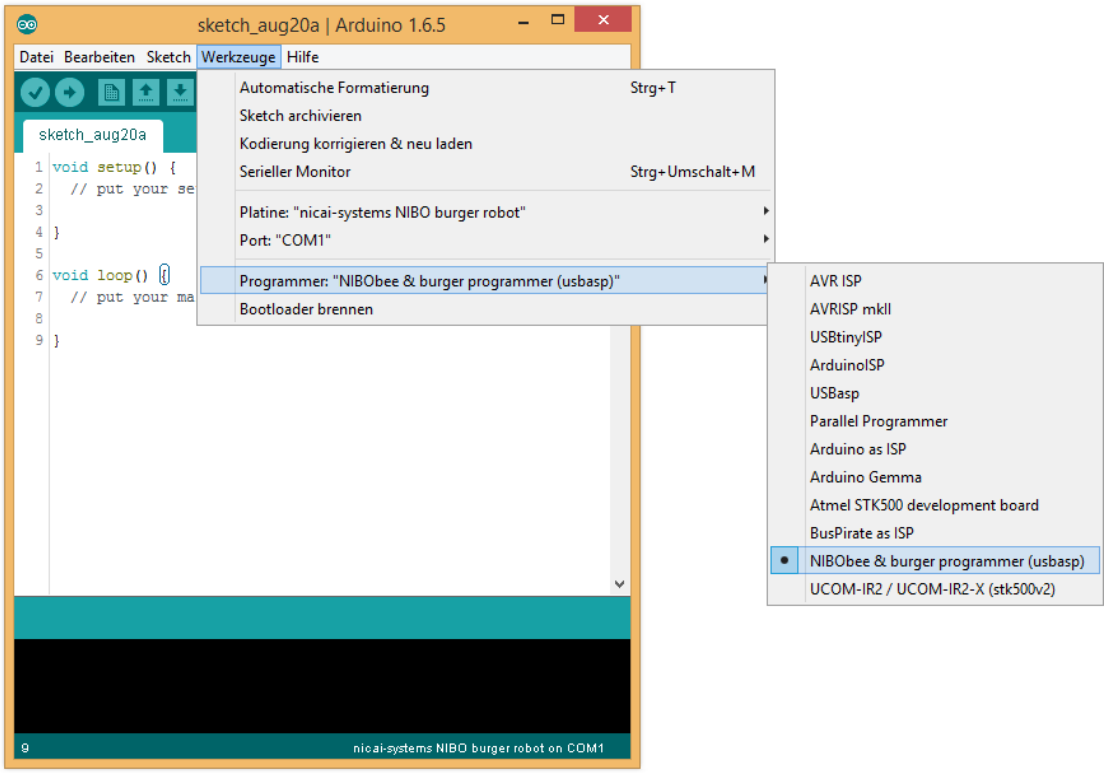




With the menu item ***Tools – Board – nicai-systems NIBO burger robot*** you must select the NIBO burger as board:

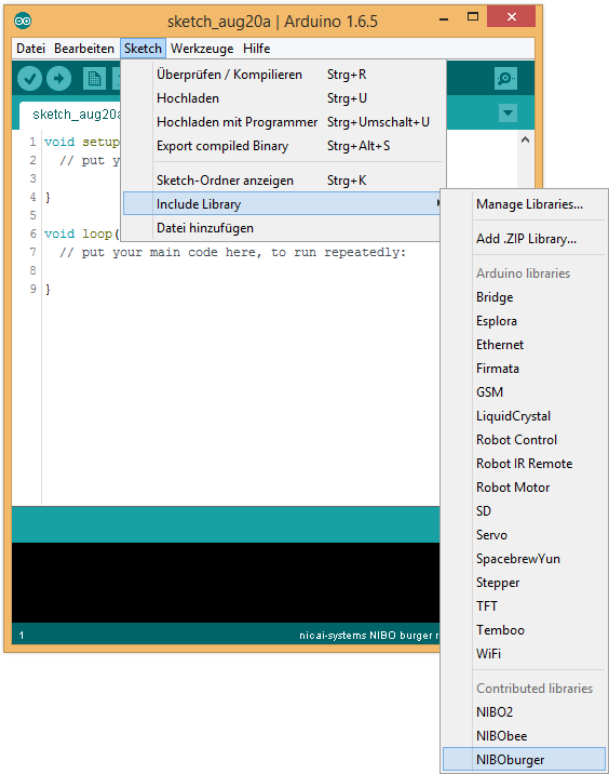


With the menu item ***Tools – Programmer – NIBObee & burger programmer*** you must select the integrated programmer of the NIBO burger as programmer:

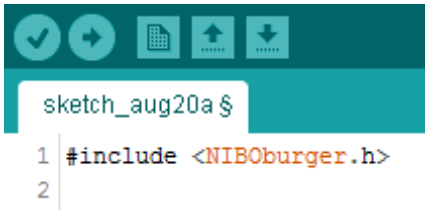


5 The first Sketch

First of all we have to import the „NIBOburger“ library:

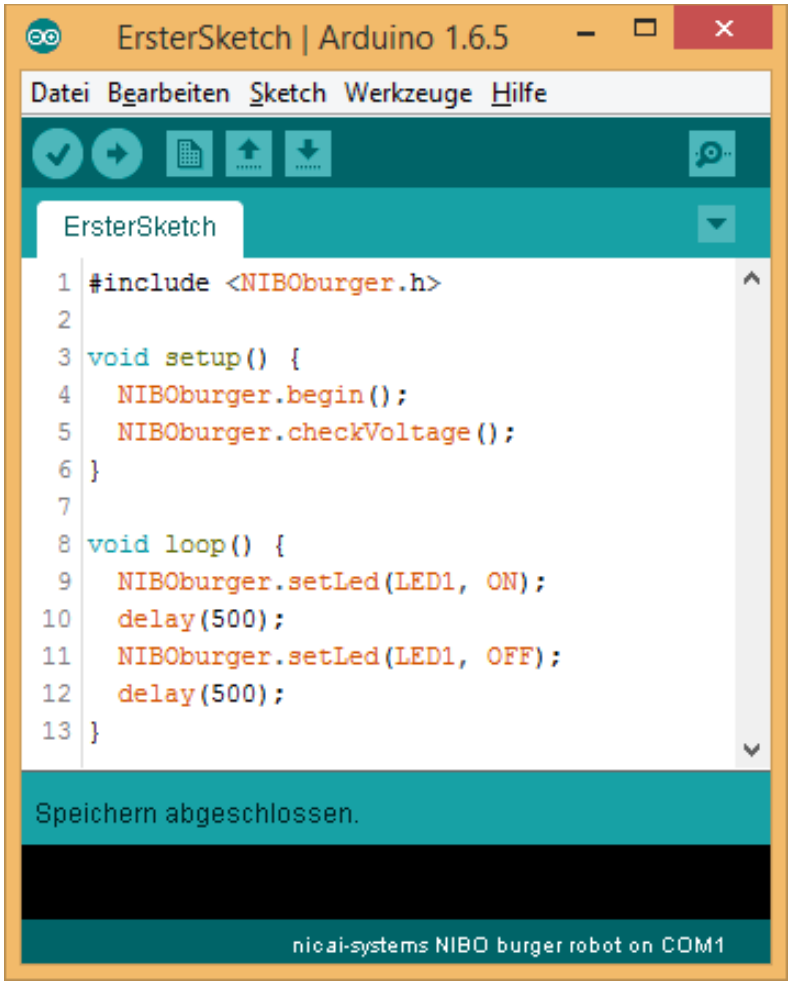


The import of the library generates the following program code:



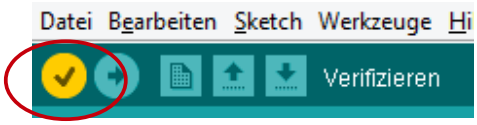
Save your sketch with the name „***FirstSketch***“.

Now complete the code as shown here:



Save your completed program again!

Now the program gets verified by click on the ***Verifying-Button***:

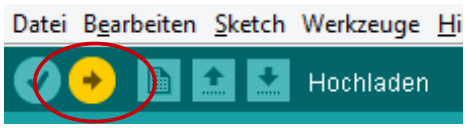


The sketch will now be compiled and verified. All possibly appearing error messages are shown in the black area of the main window.

If everything worked well the message „*compilation finished*“ appears in the turquoise area.

Switch-on the NIBO burger and connect it to the PC.

Now you can transfer your first sketch to the robot.
Therefore press the ***Upload-Button***:



If now the left red LED (*LED1*) starts to flash, then everything worked well!

Explanations of the code:

Basic set-up of an ARDUINO-Sketch:

```
#include <Bibliothek.h>
  // with the #include-instructions the chosen libraries are included.
  // Classes, variables and constants are defined in the libraries.


void setup() {
  // all instructions between the curly brackets of the
  // setup-Block are executed exactly one time at the beginning
  // during the program sequence.
}

void loop() {
  // all instructions between the curly brackets of the
  // loop-Block are executed several times
  // during the program sequence.
}
```

Back to our first sketch:

ErsterSketch

```
1  #include <NIBOburger.h>
2
3  void setup() {
4    NIBOburger.begin();
5    NIBOburger.checkVoltage();
6  }
7
8  void loop() {
9    NIBOburger.setLed(LED1, ON);
10   delay(500);
11   NIBOburger.setLed(LED1, OFF);
12   delay(500);
13 }
```

01	`#include` instruction: the header-file of the NIBO burger library is included.
03-06	Function-block to initialize the program. The function is executed exactly one time after switching-on the robot. The instruction „**NIBOburger**.begin();“ initializes the robot. ***Important:*** This instruction **has to be** the first instruction in **every** setup-function! The instruction „**NIBOburger**.checkVoltage();“ checks the voltage of the batteries after switching-on the robot. Whenever the voltage is below the threshold, the LEDs on the NIBO burger will flash...
08-13	The main program is between the line „`void loop() {`“ and the line with the closing curly bracket „`}`“. The function is called every times until the robot is switched off.
09	Inside the main block LED 1 (red LED on left side) is switched on with the expression „**NIBOburger**.setLed (LED1, ON);“.
10	Afterwards a pause of 500 milliseconds (half second) is executed by a call to „`delay(500);`“.
11-12	After the first delay, LED 1 will be switched off. The next line executes a second delay of 500 ms.
13	After the second delay, the loop-funktion is finished and will be executed again automatically. **That's it!**

Note: All examples of this tutorial are to find here: „**File**“ → „**Examples**“ → „**NIBO burger**“ → „**Tutorial**“.

6 LEDs

Now we want to perform another experiment with the LEDs.

Create a new sketch – select „***File***“ → „***New***“ from the menu. Save the new sketch under the filename „***LEDs***“.

You must include the library „NIBOburger“ as described in chapter 5.

This time the LEDs will we switched on and off around.

Enter the following source code into the editor window:

Explanation about the source code:

01-06	These lines are the same as in chapter 5!
08-15	The first expression „`for (int led=1; led<5; led++)`“ serves to execute the expressions inbetween the curly brackets multiple times. In this case the block is executed exactly 4 times. The variable „`led`“ changes its value during every iteration: During the first iterationthe value is 1, in the following iterations the value is 2, 3 and 4.

The for-loop is a powerful general-purpose tool and consists of the following elements:

```
for (Initialization; Condition; Continuation) {
  // this block will we be executed as long as the condition is true.
}
```

09

In the initialization part of the for we declare an integer variable „led“ and initialize it with the value 1. The condition part of the loop is, that the variable „led“ is smaller than 5. As continuation we increment the value of the variable „led“ by one. The expression „led++“ is a shortcut for the expression „led = led + 1“.
Therefore the for-loop is executed step by step with the values 1, 2, 3 and 4 for the variable „led“.

10-13

What's to be done is written in the body of the loop between the curly braces:

With the expression „**NIBOburger**.setLed(led, ON);“ the LED with the number of the variable „led“ will be switched on.

The expression „delay(350);“ will wait 350 ms before the execution of the code continues.

With the expression „**NIBOburger**.setLed(led, OFF);“ the LED with the number of the variable „led“ will be switched off.

The last expression in the block will delay the execution 150 Milliseconds, before the next loop continues.

7 LEDs Action

Now we will continue with a more complex light pattern.

Create a new sketch – select „***File***“ → „***New***“ from the menu. Save the new sketch under the filename „***LEDsAction***“.

You must include the library „NIBOburger“ as described in chapter 5.

This time the LEDs will we switched on and off around. We will change the speed from turn to turn.

Enter the following source code into the editor window:

LEDsAction

```
1  #include <NIBOburger.h>
2
3  void setup() {
4    NIBOburger.begin();
5    NIBOburger.checkVoltage();
6  }
7
8  void loop() {
9    for (int time=50; time<800; time*=2) {
10     for (int led=1; led<5; led++) {
11       NIBOburger.setLed(led, ON);
12       delay(time);
13       NIBOburger.setLed(led, OFF);
14       delay(time);
15     }
16   }
17 }
```

Explanation about the source code:

01-06	These lines are the same as in chapters 5 and 6!
08-17	The first instruction of the der loop-function „`for (int time=50; time<800; time*=2)`“ executes the instructions between the curly braces several times. In the initialization part the variable „`time`“ is declared and initialized with the value 50. The condition part of the loop is, that the variable `time` is smaller than 800. As continuation we double the value of the variable `time`.

The expression „`time*=2`“ is a shortcut for the expression „`time = time * 2`“.
Therefore the outer for-loop is executed step by step with the values 50, 100, 200 and 400 for the variable `time`.

`10-15` The inner for-loop is executed for the variable `led`. The variable gets step by step the values 1 - 4.

`11-14` The real instructions are to find in the body of the inner for-loop:

With the instruction „**NIBOburger**.setLed(`led`, `ON`);“ the LED with the number **`led`** is switched-on.

The expression „delay(`time`);“ will wait a time span depending on the value of the variable `time` before the execution of the code continues.

With the instruction „**NIBOburger**.setLed(`led`, `OFF`);“ the LED with the number **`led`** is switched-off.

The last instruction of the block causes the waiting of a time span depending on the value of the variable `time` before the execution of the next loop cycle starts.

Ideas & Proposals:

1. Change your program code so that the chronological sequence is reversed. The speed should rise from turn to turn.
2. Change your program code so that the execution direction is reversed. The led LED1 shall be the last flashing led in each cycle.

8 LEDs Random

Now we will code a random light pattern...

Create a new sketch – select „***File***“ → „***New***“ from the menu. Save the new sketch under the filename „***LEDsRandom***“.

You must include the library „NIBOburger“ as described in chapter 5.

To get a random flashing we will switch-on one led and switch-off one led at each cycle.

Enter the following source code into the editor window:

LEDsZufall

```
1  #include <NIBOburger.h>
2
3  void setup() {
4    NIBOburger.begin();
5    NIBOburger.checkVoltage();
6    delay(100);
7    NIBOburger.randomize();
8  }
9
10 void loop() {
11   int led = NIBOburger.getRandomInt(1,4);
12   NIBOburger.setLed(led, ON);
13   delay(100);
14   led = NIBOburger.getRandomInt(1,4);
15   NIBOburger.setLed(led, OFF);
16   delay(100);
17 }
```

Explanation about the source code:

01-06	These lines are the same as in chapters 5 and 6!
07	The instruction „**NIBOburger**.randomize();“ initializes the random number generator. We use the last sampled sensor data (sensor bricks, power voltage and voltage of the key buttons) as initial value for the random number generator.

10-17 In each cycle a random LED is switched-on and the program waits. Then a random LED is switched-off and the program waits again.

11 The procedure call „**NIBOburger**.getRandomInt(1, 4)“ returns a random number between 1 and 4.

Ideas & Proposals:

1. Change your program code so that the waiting times for the breaks will be chosen randomized too.
2. Change your program code so that always the LED is switched-off which has been switched-on before.

9 Push-Button

Now we will test and program the buttons.

Create a new sketch – select „***File***“ → „***New***“ from the menu. Save the new sketch under the filename „***Buttons***“. We also include the library „NIBOburger“ as described in chapter 5.

Enter the following source code into the editor window:

Taster

```
1  #include <NIBOburger.h>
2
3  void setup() {
4    NIBOburger.begin();
5    NIBOburger.checkVoltage();
6  }
7
8  void loop() {
9    char key = NIBOburger.getKeyChar();
10
11   switch (key) {
12     case 'A':
13       NIBOburger.setLed(LED1, ON);
14       break;
15     case 'B':
16       NIBOburger.setLed(LED2, ON);
17       break;
18     case 'C':
19       NIBOburger.setLed(LED3, ON);
20       break;
21     case 'a':
22       NIBOburger.setLed(LED1, OFF);
23       break;
24     case 'b':
25       NIBOburger.setLed(LED2, OFF);
26       break;
27     case 'c':
28       NIBOburger.setLed(LED3, OFF);
29       break;
30   }
31 }
```

Our program shall do the following:

If button 1 is pressed the LED 1 shall flash.

If button 2 is pressed the LED 2 shall flash.

If button 3 is pressed the LED 3 shall flash.

A single keypress sends two events:
First we get a capital letter A-C for the pressing of the button, and afterwards we get a small letter a-c for the releasing of the button.

Explanation about the source code:

01-06	These lines are the same as in the chapters 5 and 6!
	loop-function:
09	In this line we define the variable „key“. This variable saves the last keyboard action, which we will get by calling the procedure „**NIBOburger**.getKeyChar()“. The procedure call returns one of the following values: 0: nothing has changed since the last call. 'A': Button 1 was pressed. 'a': Button 1 was released. 'B': Button 2 was pressed. 'b': Button 2 was released. 'C': Button 3 was pressed. 'c': Button 3 was released.
11-30	The **switch**-instruction in line 10 provides a distinction of cases. Depending on the value of the variable „key“, a special case will be executed. The single cases are initiated with the keyword **case**. They are stopped with a **break**-instruction. The switch-instruction executes the different instruction-blocks depending on the value of the varaible „key“:
12-14	If „key“ has the value '**A**', the LED 1 will be switched-on.

```
15-20   Equivalent for 'B' and 'C'...
21-23   If „key“ has the value 'a', the LED 1 will be switched-off.
24-29   Equivalent for 'b' and 'c'...
```

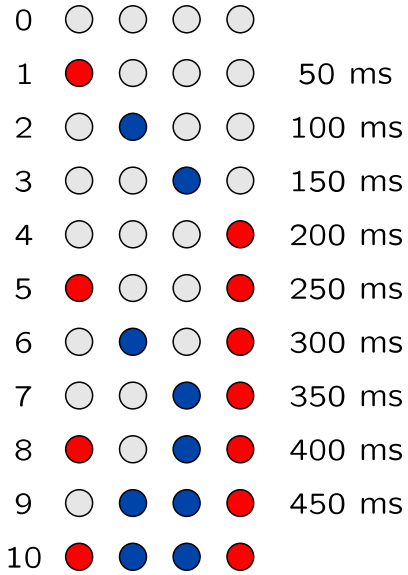
Ideas & Proposals:

1. Change your program code so that the respective LED will be switched-on by a first push of the button, and it shall be switched-off by a second push.

10 Reaction

In this example we will write a small „game“: The program should measure the reaction time and display how fast humans may react.

The four LEDs are used to display the passed time using the following schema:



The game is started by pressing **Key 1** – afterwards a random timespan between one and four seconds follows.

When the LEDs begin to flash, you must press a key as fast as possible. The resulting LED pattern stands for the reaction time as seen in the table obove.

If all LEDs are glowing you pressed to early, to late or even not at all!

We define two functions for the game:

The function waitForKey() will wait a defined time for a key press event. If any key was pressed during the interval the function will return true. If no key was pressed, the function returns false. The datatype for the logical value is called **bool** and can have the values **true** and **false**.

The function displayState() displays the current state (0-10) with the LEDs as described in the table above.

Create a new sketch as in the preceding examples and save it under the filename „***Reaction***“. Import the library „NIBOburger“ once again.

Complete the sourcecode in the editor window:

```
01 #include <NIBOburger.h>
02
03 bool waitForKey(unsigned int millisec) {
04   for (unsigned int t=0; t<millisec; t++) {
05     if (NIBOburger.getKeyChar()!=0) {
06       return true;
07     }
08     delay(1);
09   }
10   return false;
11 }
12
13 void displayState(unsigned int state) {
14   switch (state) {
15     case 0:  NIBOburger.setLeds(OFF, OFF, OFF, OFF); break;
16     case 1:  NIBOburger.setLeds( ON, OFF, OFF, OFF); break;
17     case 2:  NIBOburger.setLeds(OFF,  ON, OFF, OFF); break;
18     case 3:  NIBOburger.setLeds(OFF, OFF,  ON, OFF); break;
19     case 4:  NIBOburger.setLeds(OFF, OFF, OFF,  ON); break;
20     case 5:  NIBOburger.setLeds( ON, OFF, OFF,  ON); break;
21     case 6:  NIBOburger.setLeds(OFF,  ON, OFF,  ON); break;
22     case 7:  NIBOburger.setLeds(OFF, OFF,  ON,  ON); break;
23     case 8:  NIBOburger.setLeds( ON, OFF,  ON,  ON); break;
24     case 9:  NIBOburger.setLeds(OFF,  ON,  ON,  ON); break;
25     case 10: NIBOburger.setLeds( ON,  ON,  ON,  ON); break;
26   }
27 }
28
29 void setup() {
30   NIBOburger.begin();
31 }
32
33 void loop() {
34   NIBOburger.checkVoltage();
35   delay(100);
36   while (NIBOburger.getKeyChar()!='A') {
37     // auf KEY-DOWN von Taster 1 warten....
38   }
39
40   displayState(0);
41   while (NIBOburger.getKeyChar()!='a') {
42     // auf KEY-UP von Taster 1 warten....
43   }
44
45   NIBOburger.randomize();
46   unsigned int time = NIBOburger.getRandomInt(1000, 4000);
47
48   if (waitForKey(time)) {
49     displayState(10);
50     return;
51   }
52
53   for (unsigned int i=1; i<=10; i++) {
```

```
54      displayState(i);
55      if (waitForKey(50)) {
56        return;
57      }
58    }
59  }
```

Explanation about the source code:

03	The function **`waitForKey`**() gets the time to wait for a key press as parameter.
04	The for loop will be executed exactly once per millisecond.
05-07	If a key was pressed the function will return with the value **true**.
10	After the time has elapsed the function will return with the value **false**.

13	The function **`displayState`**() gets the status to display as parameter. It has no return value (**void**).
14-26	Depending on the state, the four LEDs will be switched with the relevant call to **NIBOburger**.setLeds().

29-31	**`setup`-function:** The setup-function begins as always with the initialization.

33-59	**`loop`-function**
34-35	In the beginning the supply voltage will be checked and a short delay will happen.
36-38	The while loop will execute until Key 1 is pressed down.
40-43	The LEDs will be switched off. The consequent while loop executes until Key 1 is released.
45-46	The random number generator is initialized by random sensor data, and generates a number between 1000 and 4000.
48-51	We will wait the random time for a key press. If the key press happens it was to early, status 10 will be displayed and we return from the loop function.

53	The loop will be executed ten times with the following values: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
54-57	With a call to **`displayState`**(), the current status is displayed. If a key was pressed within 50 milliseconds, we are finished and return from the loop function.

Ideas & Proposals:

1. Modify your program in the way, that it measures in 500 ms steps. In this way you can observe the single steps much better.
2. Modify your program in the way, that it measures in 20 ms steps. The maximum reaction time is reduced to 180 ms...

11 Test of Odometry Sensors

In this sketch we will test the odometry sensors.

Create a new sketch „***Odometry***“ and import the library „NIBOburger“.

This program displays the values of the odometry counters with the LEDs:
If the counter is above 10 the red LED will glow, if the value is above 20 the blue LED will glow also.

By pressing the keys the counters can be reset:

Key 1 resets the left counter,
Key 2 resets the right counter,
Key 3 resets both counters.

Enter the following source code into the editor window:

Odometrie

```
#include <NIBOburger.h>

void setup() {
  NIBOburger.begin();
}

void loop() {
  NIBOburger.checkVoltage();

  char key = NIBOburger.getKeyChar();
  if (key=='a') {
    Engine.left.resetTicks();
  }
  if (key=='b') {
    Engine.right.resetTicks();
  }
  if (key=='c') {
    Engine.left.resetTicks();
    Engine.right.resetTicks();
  }

  NIBOburger.setLed(LED1, Engine.left.getTicks()>10);
  NIBOburger.setLed(LED2, Engine.left.getTicks()>20);
  NIBOburger.setLed(LED3, Engine.right.getTicks()>20);
  NIBOburger.setLed(LED4, Engine.right.getTicks()>10);
}
```

Explanation about the source code:

01-06	The program begins with the include of the library and the initialization of the robot.
07-26	**`loop`-Funktion:**
08	Monitor the supply voltage.
09	The current key event is stored in the variable `key`.
11-20	The three if-statements will reset the odometry counters if a key was released.
22-25	The last four lines display the odometry counter values with the LEDs. The call to `Engine.left.getTicks()` returns the current odometry counter without resetting it. ***Hint:*** The odometry counter will only increment in thois example, independent of the turn direction. If the motor is used to turn the wheel, the polarity determines the direction of counting.

Ideas & Proposals:

Modify the program in the following way:

- Define a global variable `threshold` and initialize it with the value 5.
- If you press key 1 `threshold` should be incremented by one.
- If you press key 3 `threshold` should be decremented by one.
- If you press key 2 the odometry counters should be reset.
- The blue LEDs should glow, if the counter is obove the `threshold`, the red LED should glow if its below...

12 Motor Control

Now we will control the motors.

Create a new sketch „***Motor***“ and import the library „NIBOburger“.

The program will control the motors with the buttons. If you press button 1, the motors will move forward. If you press button 3, the motors will move backward.

Button 2 is to stop both motors. The program shall react on the **push** of the button (capital letter **B**).

The program shall react on the **release** of button 1 and button 3 (small letters **a** and **c**).

Enter the following source code into the editor window:

Motor

```
#include <NIBOburger.h>

void setup() {
  NIBOburger.begin();
}

void loop() {
  NIBOburger.checkVoltage();
  int speed = 0;
  char c = NIBOburger.getKeyChar();

  if (c!=0) {
    switch (c) {
      case 'a': speed = 800; break;
      case 'B': speed = 0; break;
      case 'c': speed = -800; break;
    }
    Engine.setPWM(speed, speed);
  }

  delay(10);
}
```

Explanation about the source code:

01-06	The program starts with the including of the NIBOburger library and the initialization of the robot.
08-22	**loop-function:**
09-10	The local variable „speed“ is declared. We save the set point for the motors in this variable. At first the variable gets the value 0.
11-20	The set point for the motors is set in the switch-instruction block depending on the buttons.
18	In this line the set point is set for both motors.

Note:

The declaration of the values is given in 1/1024 steps:
+1024 means 100% forward,
+ 512 means 50% forward,
- 512 means 50% backward.
0 means stop
The control is based on pulse duration modulation (PDM).

Ideas & Proposals:

1. Modify the program in the way that the forward movement is faster and the backward movement is slower.
2. Modify the program in the way that a first keypress changes the power to 50% and a second keypress changes the power to 100%.

13 Query the Sensors

In this example we will analyse the measurement values from the sonsors.

Create a new sketch „***Sensor***“ and import the library „NIBOburger“.

The program will display the values of the two outer front sensor slots **FLL** (left) und **FRR** (right) with the LEDs: Whenever the value is higher than 15, the red LED will glow, if the value is higher then 25 the blue LED will glow also.

Enter the following source code into the editor window:

Sensoren

```
#include <NIBOburger.h>

void setup() {
  NIBOburger.begin();
}

void loop() {
  NIBOburger.checkVoltage();
  unsigned int l = SensorFLL.get();
  unsigned int r = SensorFRR.get();

  NIBOburger.setLed(LED1, l>15);
  NIBOburger.setLed(LED2, l>25);
  NIBOburger.setLed(LED3, r>25);
  NIBOburger.setLed(LED4, r>15);

  delay(10);
}

```

Explanation about the source code:

01-05	The program starts with the including of the NIBOburger library and the initialization of the robot.
07-19	**`loop`-function:**
08	Check the batteries...
09-10	The values of the sensors will be stored in the two local variables **`l`** and **`r`**.
12-15	The LEDs will be switched on and off dependent on the values.

Ideas & Proposals:

1. Use the inner front sensors (**FL** und **FR**) instead of the outer ones.
2. Modify the threshold values. Which interval for the sensor values makes sense?

14 Obstacle Detection

Now NIBO burger shall detect and avoid obstacles.

Create a new sketch „***Obstacle***“ and import the library „NIBOburger“.

The NIBO burger drives autonomous with this program: As soon as the sensors detect an obstacle, the robot shall try to avoid it.

Enter the following source code into the editor window:

```
01 #include <NIBOburger.h>
02
03 enum {
04   OBST_CLEAR = 0,
05   OBST_LEFT  = 1,
06   OBST_RIGHT = 2,
07   OBST_BOTH  = 3
08 };
09
10 unsigned int environment;
11
12 unsigned int calculateEnvironment() {
13   unsigned int left = max(SensorFLL.get(), SensorFL.get());
14   unsigned int right = max(SensorFRR.get(), SensorFR.get());
15
16   if ((left>15)&&(right>15)) {
17     return OBST_BOTH;
18   }
19   if (left>15) {
20     return OBST_LEFT;
21   }
22   if (right>15) {
23     return OBST_RIGHT;
24   }
25   return OBST_CLEAR;
26 }
27
28 void setup() {
29   NIBOburger.begin();
30   environment = OBST_CLEAR;
31 }
32
33 void loop() {
34   NIBOburger.waitAnalogUpdate();
35   NIBOburger.checkVoltage();
36   unsigned int new_env = calculateEnvironment();
37
38   if (new_env!=environment) {
39     environment = new_env;
```

```
40      switch (new_env) {
41        case OBST_CLEAR:
42          NIBOburger.setLed(LED2, 0);
43          NIBOburger.setLed(LED3, 0);
44          Engine.setSpeed(+25, +25);
45          break;
46
47        case OBST_LEFT:
48          NIBOburger.setLed(LED2, 1);
49          NIBOburger.setLed(LED3, 0);
50          Engine.setSpeed(+20, -20);
51          break;
52
53        case OBST_RIGHT:
54          NIBOburger.setLed(LED2, 0);
55          NIBOburger.setLed(LED3, 1);
56          Engine.setSpeed(-20, +20);
57          break;
58
59        case OBST_BOTH:
60          NIBOburger.setLed(LED2, 1);
61          NIBOburger.setLed(LED3, 1);
62          Engine.setSpeed(0, 0);
63          break;
64      }
65    }
66 }
```

Explanation about the source code:

01-23	**Declarations**
03-08	In this lines some constants for the „conditions of the robots environment“ are defined.
10	The global variable `environment` saves the status of the last loop of the loop-function.
12-26	The function calculateEnvironment takes the sensordata and calculates the current status ov the environment in front of the roboter.
13+14	The function max returns the maximum of two values. It is used to calculate the maximum sensor signal on each side.
16-25	The function returns OBST_LEFT if there is a close object on the left side. It returns OBST_RIGHT if there is a close object on the right side. If there are close objects on both sides the function returns OBST_BOTH. If there are no objects in front of the robot the function returns OBST_CLEAR.

28-30 **setup-function:**

The setup-function begins as always with the initialization. In the next line the variable environment is initialized to the value OBST_CLEAR.

33-66 **loop-function:**

34-36 We wait for an update of the sensor data, check the supply voltage and calculate the current state of the environment in front of the roboter.

38 The following block is executed only when the current state changes (`new_env` differs from `environment`).

40-64 Dependend on the new state, the motors power is controlled and the LEDs are switched.

Ideas & Proposals:

1. Extend the program with a new state OBST_CLOSE, which stands for a close object in front of the roboter. In this case the robot should move backward slowly.
2. Improve and extend the program!

15 Calibration of the Colour Sensor

In this example we will deal with the colour sensor of the NIBO burger. The colour sensor is build up of three colour LEDs (red, green and blue) and three phototransistors. To use the colour sensor it is neccesary to calibrate it first.

Therefore enter the following source code into the editor window:

Kalibrierung

```
#include <NIBOburger.h>

void setup() {
  NIBOburger.begin();
}

void loop() {
  NIBOburger.waitAnalogUpdate();
  NIBOburger.checkVoltage();

  ColorRGB col = SurfaceSensor.getColorRGB();
  int diff_black = col.diff(ColorRGB::CALIBRATE_BLACK);
  int diff_white = col.diff(ColorRGB::CALIBRATE_WHITE);

  NIBOburger.setLed(LED1, diff_black < 20);
  NIBOburger.setLed(LED2, diff_white < 20);

  char key = NIBOburger.getKeyChar();
  if (key=='a') {
    SurfaceSensor.calibrateBlack();
    SurfaceSensor.storeCalibration();
  } else if (key=='b') {
    SurfaceSensor.calibrateWhite();
    SurfaceSensor.storeCalibration();
  }
}
```

After transferring the software the sensors can be calibrated as follows:

- place the NIBO burger with all three surface sensors on a **black area** and press **button 1**.
- place the NIBO burger with all three surface sensors on a **white area** and press **button 2**.
- The calibration data is stored automatically.

To generate the sensor data, the IR-reflection method is used. Therefor the reflected fraction from the emitted light is measured. The measurement takes place two times: One time with the LED switched on and one time with the LED switched off. By substraction of the two values the influence of ambient light can be minimized.

After calibration the measured values will be normalized by the library and they are available as integer numbers in the range from **0 (black)** bis **1024 (white/coloured)** by use of the following expressions:

```
SensorBL.get()
SensorBC.get()
SensorBR.get()
```

Alternatively the values are also available as RGB and HSV color values:

```
SurfaceSensor.getRGB()
SurfaceSensor.getHSV()
```

The calibration parameters are stored permanently in the EEPROM. If you reprogram the Atmel ATmega16, the calibration data will survive the procedure!

If you use the IR-sensor bricks in the surface slots instead of the colour bricks, you must recalibrate the sensors!

16 Using the Colour Sensor

Now we can use the colour sensor:

The testprogram should display with the LEDs on which area of the colourcard (blue, red, green oder yellow) the roboter is placed.

Create a new sketch, and save it unter the file name „**Coloursensor**“. Import the library „NIBOburger“ and complete the sourcecode in the editor window:


```
#include <NIBOburger.h>

void setup() {
  NIBOburger.begin();
}

void loop() {
  NIBOburger.waitAnalogUpdate();
  NIBOburger.checkVoltage();

  ColorHSV col = SurfaceSensor.getColorHSV();
  int diff_blue   = col.diff(ColorHSV::CALIBRATE_BLUE, 20);
  int diff_red    = col.diff(ColorHSV::CALIBRATE_RED, 20);
  int diff_green  = col.diff(ColorHSV::CALIBRATE_GREEN, 20);
  int diff_yellow = col.diff(ColorHSV::CALIBRATE_YELLOW, 20);

  int diff_min = min(min(diff_blue, diff_red), min(diff_green, diff_yellow));

  if (diff_min<600) {
    if (diff_min==diff_blue) {
      NIBOburger.setLeds(0, 1, 1, 0);
    } else if (diff_min==diff_red) {
      NIBOburger.setLeds(1, 0, 0, 1);
    } else if (diff_min==diff_green) {
      NIBOburger.setLeds(1, 1, 0, 0);
    } else if (diff_min==diff_yellow) {
      NIBOburger.setLeds(0, 0, 1, 1);
    }
  } else {
    NIBOburger.setLeds(0, 0, 0, 0);
  }
}
```

Explanation about the source code:

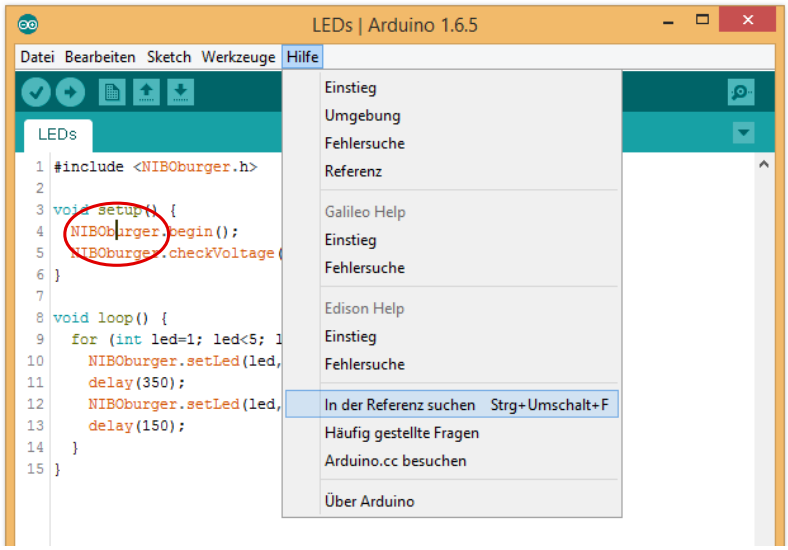
01-05	The program starts with the including of the NIBOburger library and the initialization of the robot.
08-13	**loop-function:**
08-09	We wait for an update of the sensor data and check the supply voltage.
11	This line initializes the variable col with the current colour value out of the HSV-colour space.
12-15	Now we calculate the difference between the measured colour and the 4 reference colours from the colour card.
17	We determine the minimum difference.
19	If the minimum difference is below the threshold, we detected a colour from the colour card.
20-28	Dependent on which difference was the smallest one, we switch the LEDs
29-31	Otherwise we did not recognize a colour and switch off all LEDs.

Ideas & Proposals:

1. Try to change from the HSV colour space to the RGB colour space – which one is better for the purpose?
2. Inform yourself about the RGB and HSV colour space in wikipedia, wich advantage offers the HSV colour space for surface colour detection?

17 Documentation

You can open an overview of the NIBOburger ARDUINO library: Just click on the word NIBOburger in the editor (see picture) and choose the menu item „**Help**“ → „**Find in Reference**“:



Now your webbrowser opens the following window:

NIBO burger robot class reference for **ARDUINO**

NIBO burger • **ARDUINO**-Referenz

Einbindung durch: **#include <NIBOburger.h>**

- class instance **NIBOburger**

Funktion	Beschreibung
void **begin** ()	Initialisierung des NIBOburger Roboters
unsigned int **getVoltage** ()	Versorgungsspannung messen. *return-value*: Spannung in Millivolt (4.8V ≙ 4800)
void **checkVoltage** ()	Versorgungsspannung überprüfen, im Fehlerfall anhalten und blinken
void **setLed** (int *led*, int *value*)	LED ein/ausschalten: *led*: **LED1**=1, **LED2**=2, **LED3**=3, **LED4**=4 *value*: **OFF**=0, **ON**=1
int **getLed** (int *led*)	LED abfragen: *led*: **LED1**=1, **LED2**=2, **LED3**=3, **LED4**=4 *return-value*: **OFF**=0, **ON**=1
unsigned int **getRandomSeed** ()	Zufallszahlen-Basis liefern
unsigned int **getAnalog** (unsigned char *adc_channel*, unsigned char *active*)	Rohwert eines analogen Kanals auslesen

+ class instance **Engine**

+ class instances **Engine.left / Engine.right**

+ class instance **SurfaceSensor**

+ class **ColorRGB**

+ class **ColorHSV**

+ class instances **SensorFL**, **SensorFR**, **SensorFLL**, **SensorFRR**, **SensorBL**, **SensorBC**, **SensorBR**

+ IO-Definitionen

nicai-systems website

18 Links

In this subsection you can find a selection of links to web pages with related topics.

Development environments:

Atmel: http://www.atmel.com
Web page of the microcontroller manufacturer.
There are data sheets, application notes and the development environment AVRStudio.

WinAVR: http://winavr.sourceforge.net/
AVR-GCC compiler for Windows with many add ons, especially for AVRStudio.

AVRDude: http://savannah.nongnu.org/projects/avrdude/
free programmer software (suits for the NIBO robots).

Roboter.CC: http://www.roboter.cc
Online code compiler & robotic online community, especially for robotic projects with lots of examples and user forum.

Further informations:

- **Nibo mainpage**: http://nibo.nicai-systems.de
  NIBO manufacturers web page. Provides technical information, the construction manual and additional links.
- **Nibo Wiki**: http://www.nibo-roboter.de
  provides all information about the NIBO robots.
- **Mikrocontroller**: http://www.mikrocontroller.net
  information about microcontroller and their coding.
- **AVRFreaks**: http://www.avrfreaks.net
  information about the AVR.