

KILOBOT

user manual



version 1.0
DECEMBER 2011

Documentation Author

Julien Tharin
K-Team S.A.
Rue Galilée 9, Y-Parc
1400 Yverdon-les-Bains
Switzerland

Email: info@k-team.com

Url: www.k-team.com

Documentation version

Version	Date	Author	Description
1.0	12.12.2011	J. Tharin	First draft

Trademark Acknowledgements:

IBM PC : International Business Machines Corp.
Macintosh : Apple Corp.
SUN Sparc-Station : SUN Microsystems Corp.
LabVIEW : National Instruments Corp.
Matlab : MathWorks Corp.
Webots : Cyberbotics Ltd
Logitech : Logitech Int. SA
Gumstix : Gumstix Inc.
Khepera : K-Team SA

LEGAL NOTICE:

- The contents of this manual are subject to change without notice
- All efforts have been made to ensure the accuracy of the content of this manual. However, should any error be detected, please inform K-Team.
- The above notwithstanding, K-Team can assume no responsibility for any error in this manual.

TABLE OF CONTENTS

1. INTRODUCTION.....	1
1.1 HOW TO USE THIS HANDBOOK	1
1.2 SAFETY PRECAUTIONS	2
1.3 RECYCLING.....	2
1.4 SPECIFICATIONS	3
2. UNPACKING AND INSPECTION	4
2.1 PACKAGE CONTENTS.....	4
2.2 INSPECTION.....	5
3. DESCRIPTION.....	6
3.1 OVERVIEW	6
3.2 KILOBOT HARDWARE.....	7
4. USAGE.....	8
4.1 REQUIRED HARDWARE / SOFTWARE.....	8
4.1.1 REQUIRED HARDWARE:	8
4.1.2 REQUIRED SOFTWARE:.....	8
4.2 CONNECTIONS.....	9
4.3 POWER-UP.....	10
4.4 INSTALLATION	11
4.4.1 OVERHEAD CONTROLLER (OHC) SOFTWARE INSTALLATION.....	11
4.4.2 OVERHEAD CONTROLLER (OHC) HARDWARE INSTALLATION	11
4.4.3 KILOBOT BOOTLOADER PROGRAMMING	12
4.5 USAGE	14
4.5.1 PROGRAMMING A KILOBOT GROUP WITH A NEW PROGRAM.....	14
4.5.2 OHC USE.....	15
4.5.3 KILOBOTS API	18
5. ANNEXES.....	20
5.1 EXAMPLES CODE	20
5.1.1 TRANSMITS DATA TO NEIGHBORS, AND BLINKS LED WHEN THE MESSAGE IS RECEIVED:.....	20
5.1.2 NON-BLOCKING TIMED MOVEMENT:	20
5.1.3 RANDOM MOVE IN SEE NEIGHBORS, STOP ELSE.....	22
6. WARRANTY.....	25

1. INTRODUCTION

Thank you for buying the Kilobot!

Kilobots are low cost robots designed at Harvard University's Self-Organizing Systems Research Lab <http://www.eecs.harvard.edu/ssr>. The robots are designed to make testing collective algorithms on hundreds or thousands of robots accessible to robotics researchers. Though the Kilobots are low-cost, they maintain abilities similar to other collective robots. These abilities include differential drive locomotion, on-board computation power, neighbor-to-neighbor communication, neighbor-to neighbor distance sensing, and ambient light sensing. Additionally they are designed to operate such that no robot requires any individual attention by a human operator. This makes controlling a group of Kilobots easy, whether there are 10 or 1000 in the group.

1.1 How to use this handbook

This handbook introduces the Kilobot and its various operating modes. For a quick start, jump to chapter 4 "*Usage*".

If this handbook does not answer one of the problems you wish to solve, please consult the K-Team web site (<http://www.k-team.com>) and especially the Forum and the FAQs.

- **Unpacking and Inspection** : Kilobot package description and first use.
- **Description** : Kilobot description.
- **Usage** : Kilobot usage descriptions.
- **Annexes** : Examples of source code.

1.2 Safety precautions

Here are some recommendations on how to correctly use the Kilobot:

- **Keep the board away from wet area.** Contact with water could cause malfunction and/or breakdown.
- **Store your board in a stable position.** This will avoid the risks of falling, which could break it or cause damage to a person.
- **Do not plug any connectors while the board is powered on.** To avoid any damage, make all connections when the board power is off.

1.3 Recycling

Think about the end of life of your product! Parts of the board can be recycled and it is important to do so. By recycling you can help to create a cleaner and safer environment for generations to come. For those reasons please take care to the recycling of your product at the end of its life cycle, for instance sending back the product to the manufacturer or to your local dealer.

Thanks for your contribution to a cleaner environment!

1.4 Specifications

The main specifications of the Kilobot robot are listed below:

- Processor : ATmega 328 (8bit @ 8MHz)
- Memory :
 - 32 KB Flash used for both user program and bootloader
 - 1KB EEPROM for storing calibration values and other non-volatile data and 2KB SRAM.
- Battery : Rechargeable Li-Ion 3.7V, for a 3 months autonomy in sleep mode. Each Kilobot has a built-in charger, which charges the onboard battery when +6 volts is applied to any of the legs, and GND is applied to the charging tab.
- Charging : Kilobot charger (optional)
- Communication : Kilobots can communicate with neighbors up to 7 cm away by reflecting infrared (IR) light off the ground surface.
- Sensing :
 - When receiving a message, distance to the transmitting Kilobot can be determined using received signal strength.
 - The brightness of the ambient light shining on a Kilobot can be detected.
 - A Kilobot can sense its own battery voltage.
- Movement : Each Kilobot has 2 vibration motors, which are independently controllable, allowing for differential drive of the robot. Each motor can be set to 255 different power levels.
- Light : Each Kilobot has a red/green/blue (RGB) LED pointed upward, and each color has 3 levels of brightness control.
- Software : The Kilobot Controller software is available for controlling the controller board, sending program files to the robots and controlling them.
- Programming : For programming, the open source development software WinAVR combined with AvrStudio from Amtel gives a C programming environment. An API with basic functions such as motor speed, led control, distance measurement,... is available and some examples are provided.
- Debug : A serial output header is available on each robot for debugging via computer terminal.
- Dimensions : The diameter is 33 mm and the height is 34mm (including the legs)

2. UNPACKING AND INSPECTION

2.1 Package Contents

Figure 2-1: Contents of the Kilobot Pack

Your package should contain the following items:

1. DVD-Rom with software
2. Kilobot robot

2.2 Inspection

To be continued...

3. Description

3.1 Overview

An overview of the Kilobot hardware is depicted in the Figure 3-1. The locations of various key elements are indicated for later references.

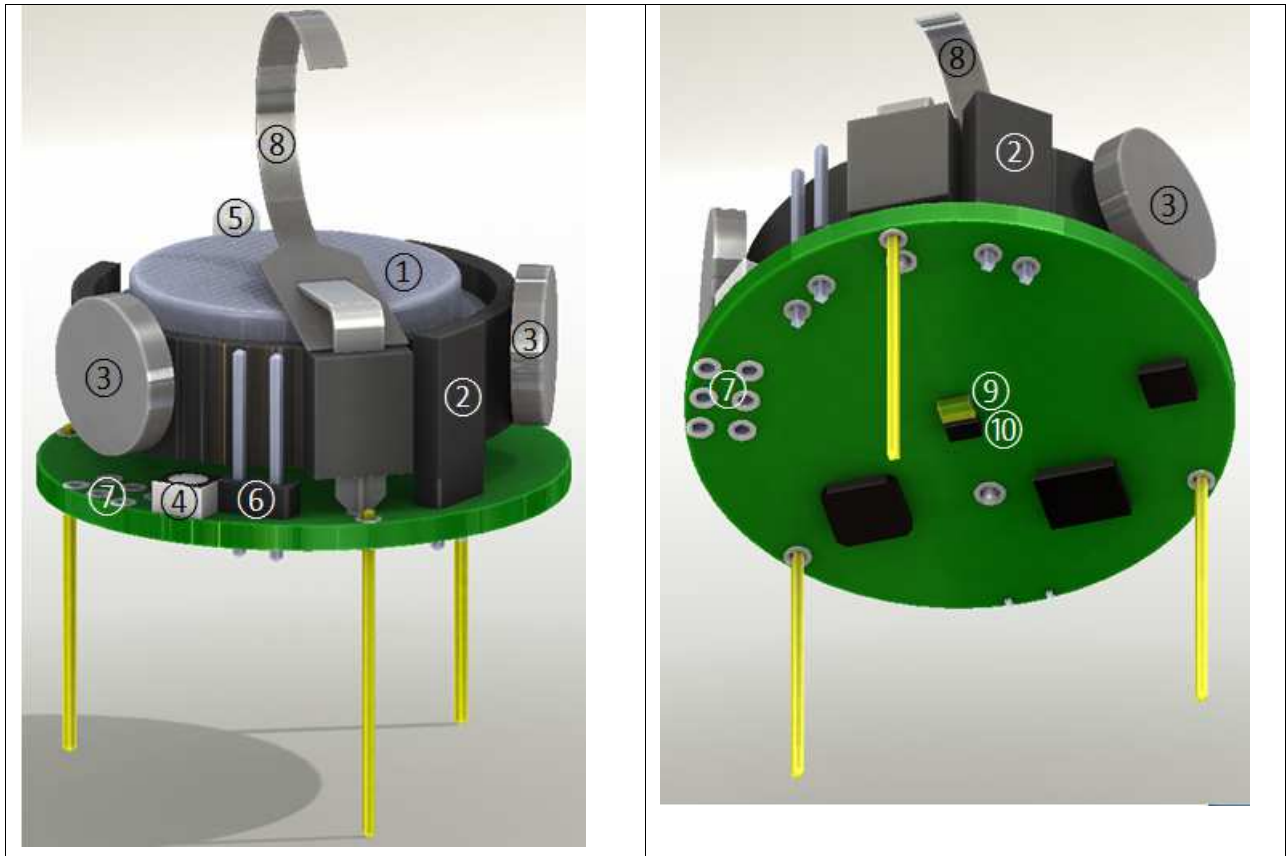


Figure 3-1: Kilobot overview

- | | |
|------------------------|-----------------------------|
| ① 3.7-Volt Battery | ⑦ Direct programming socket |
| ② Power jumper | ⑧ Charging Tab |
| ③ Vibration motors | ⑨ IR Transmitter |
| ④ LED (Red/Green/Blue) | ⑩ IR Receiver |
| ⑤ Ambient light sensor | |
| ⑥ Serial output header | |

3.2 Kilobot Hardware

The hardware of the **Kilobot** is described in the sub-chapters thereafter.

To be continued...

4. Usage

4.1 Required hardware / software

The required hardware and software to use the board and develop programs are described below.

4.1.1 Required hardware:

- Computer with Microsoft Windows with USB port (not included)
- Kilobot Robot
- over-head controller (OHC)

4.1.2 Required software:

Required files:

1. The compiler WINAVR from here:
<http://sourceforge.net/projects/winavr/files/WinAVR/20100110/>.
2. The editor AVR Studio 4 found here (requires registration):
http://www.atmel.com/forms/software_download.asp?category_id=163&family_id=607&subfamily_id=760&fn=dl_AvrStudio4Setup.exe

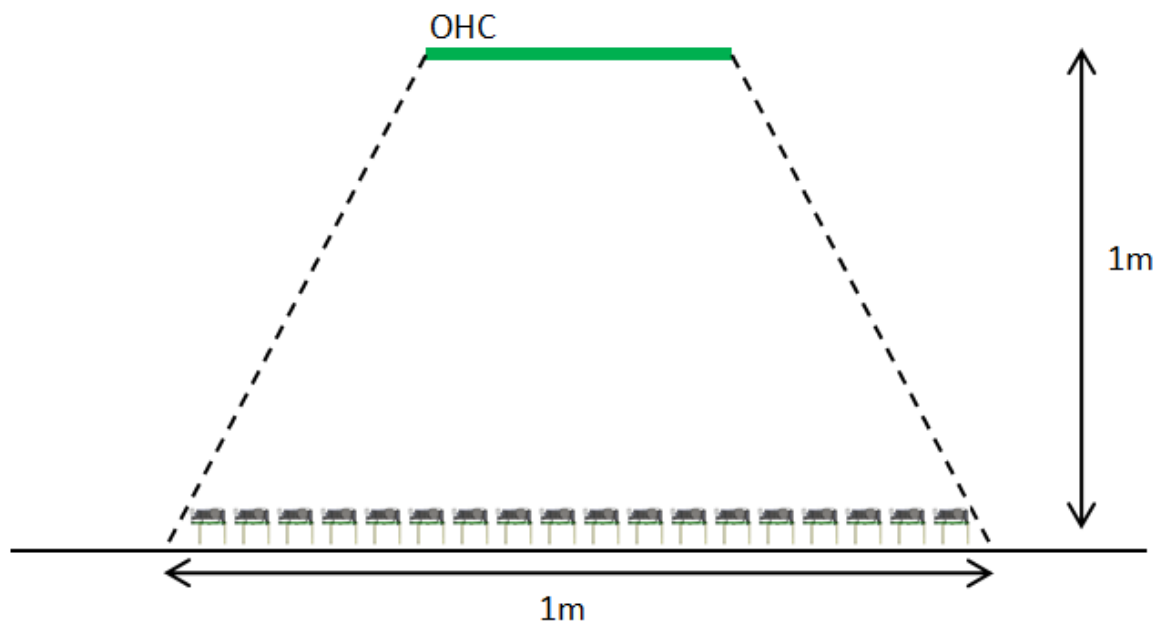
4.2 Connections

To be continued...

4.3 Power-up

Kilobots should be operated on a smooth, flat, level surface to ensure proper robot mobility. To aid communication, the surface should be glossy or reflective. A dry-erase whiteboard oriented horizontally is recommended. To prevent communication interference, Kilobots should be operated in a location out of direct sunlight or other bright sources of Infra red light.

The overhead controller should be hung above the Kilobots at a distance of about one meter. The robots beneath the OHC in a about a one meter diameter region will be able to receive messages from the OHC as shown below.



4.4 Installation

The following sub-chapters explain the software installation and the application development with the board.

4.4.1 Overhead Controller (OHC) Software Installation

1. Place *KilobotController* folder in C: (so the location is C:\KilobotController)
2. Open *KilobotController.exe*
3. In the Help menu of *KilobotController.exe*, select “Installation and programming” and follow the instructions to install the software and drivers (Fig 4.4.1).

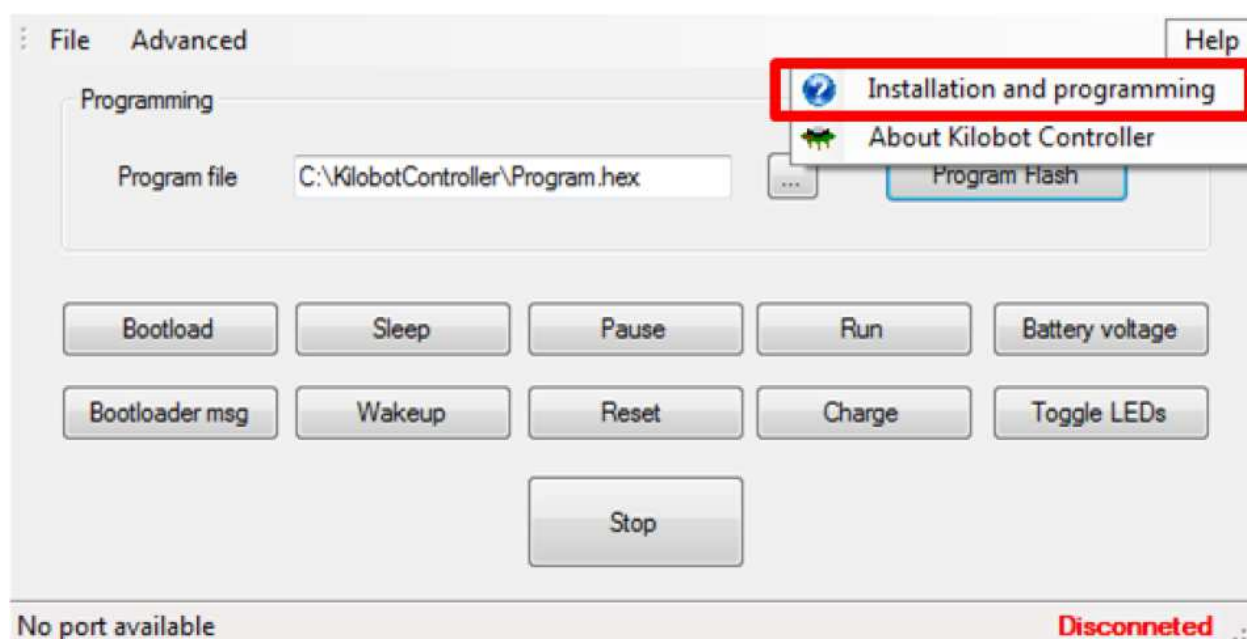


Fig 4.4.1 OHC software

4.4.2 Overhead Controller (OHC) Hardware Installation

1. Connect both USB cables to the PC.
2. Toggle the OHC power switch to turn it on.
3. Run the *KilobotController.exe* program, and click the “Toggle LEDs” button to test that a connection is made (Fig 4.4.2). This should blink the blue LEDs on the controller.

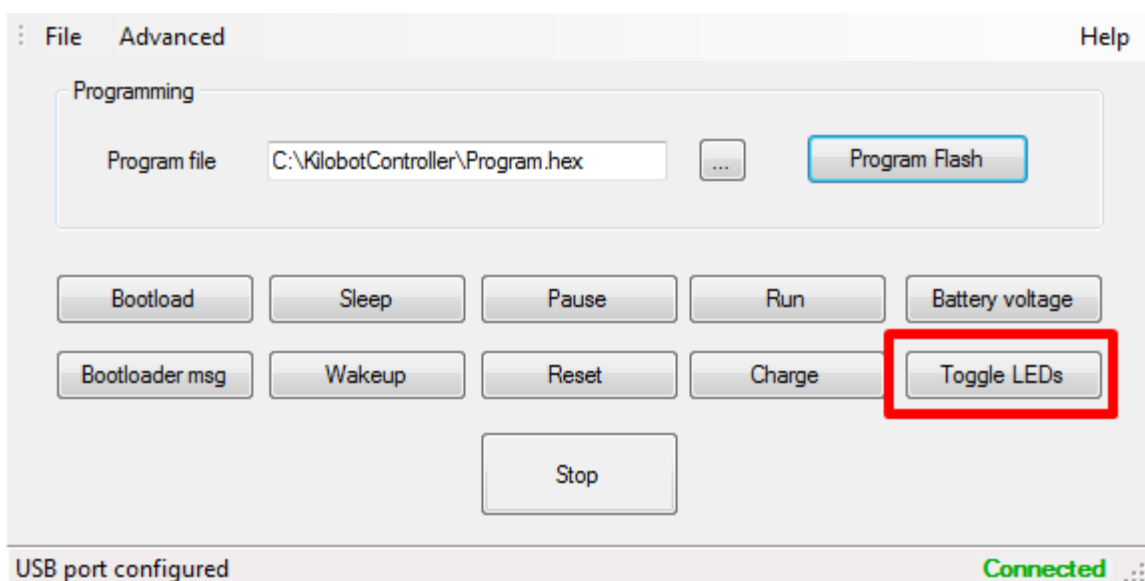


Fig 4.4.2 OHC software buttons

4.4.3 Kilobot bootloader programming

The Kilobot is delivered with a bootloader already installed. Therefore this step can be skipped. But If you would like to reinstall the bootloader or install a newer one, follow the instructions below.

1. Install WINAVR found here:
<http://sourceforge.net/projects/winavr/files/WinAVR/20100110/>.
2. and also AVR Studio 4 found here (requires registration):
http://www.atmel.com/forms/software_download.asp?category_id=163&family_id=607&subfamily_id=760&fn=dl_AvrStudio4Setup.exe
3. In AVR Studio, select Tools->ProgramAVR->Autoconnect
4. In the window that pops up, under the program tab, in the flash category select the input hex file to be *KilobotFirstFirmware.hex*.
5. Turn on the robot by adding the power jumper as shown in figure 4.4.3.

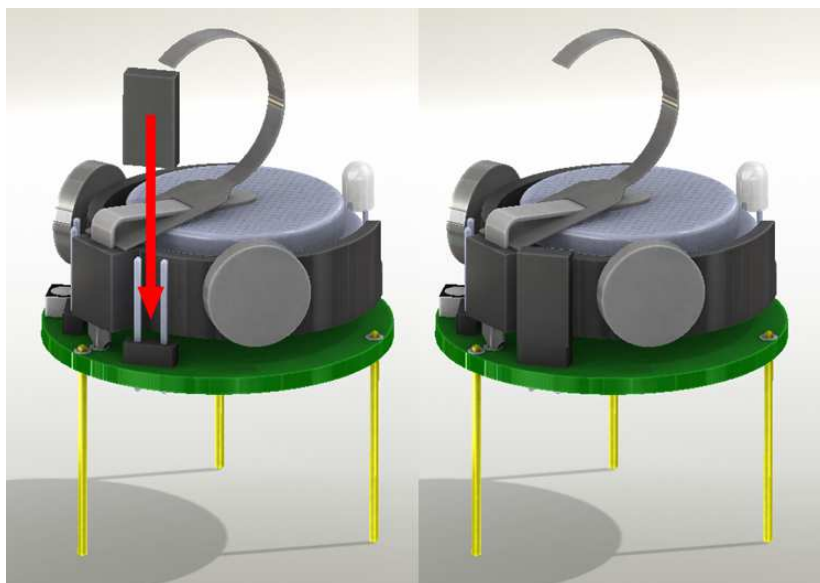


Figure 4.4.3 power jumper

6. Connect the AVRisp programmer to the robot as shown in figure 4.4.4. Make sure that the programmer pins do not touch the motor on the back side. Gently press the program cable to the side to ensure a good connection.

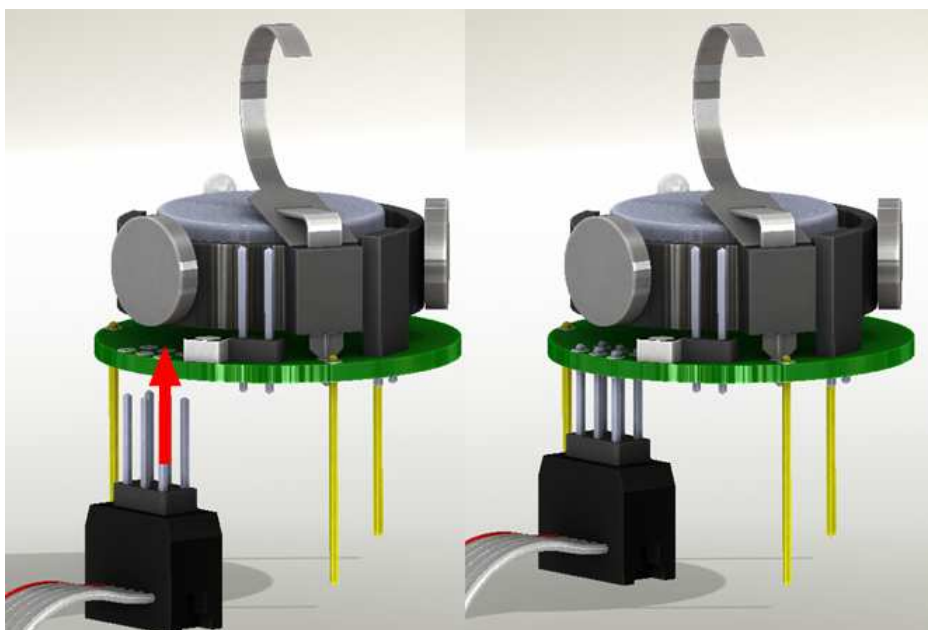


Figure 4.4.4 communication connector

7. Set robot fuses: In Avr studio AVRISP programming window, go to the fuses tab. Select only the following fuses: spien, EESAVE, BOOTSZ (boot flash size 2048 word, start address = $\$3800$) , SUT_CKSEL (int. RC Osc. 8 MHz; start-up time PWRDWN/RESET: 6 CK/14CK +65 ms). This should result in the following fuse values: extended=0xFF, high=0xD1, low=0xE2

8. In the Autoconnect pop-up window in AVRstudio press program. It should take a few seconds to program the robot, and the robot may vibrate.

4.5 Usage

4.5.1 Programming a kilobot group with a new program

1. In AVRstudio open the “Kilobot” project.
2. In *UserProgramSkeleton.c* locate the following section

```

////////////////////////////////////
//user program code goes here. this code needs to exit in a reasonable amount of time
//so the special message controller can also run
////////////////////////////////////

```

3. Immediately after this is where the user code is written. Some examples of the API are given in comments. Note that the user code should not contain long blocking functions; otherwise the robot may not respond to the overhead controller properly.
4. Once program is written, in AVRstudio select Build->build.
5. Run *KilobotController.exe*, click the button to browse for a program file as shown and select the program [.hex] file you just built (located at *kilobot/default/UserProgramSkeleton.hex*).

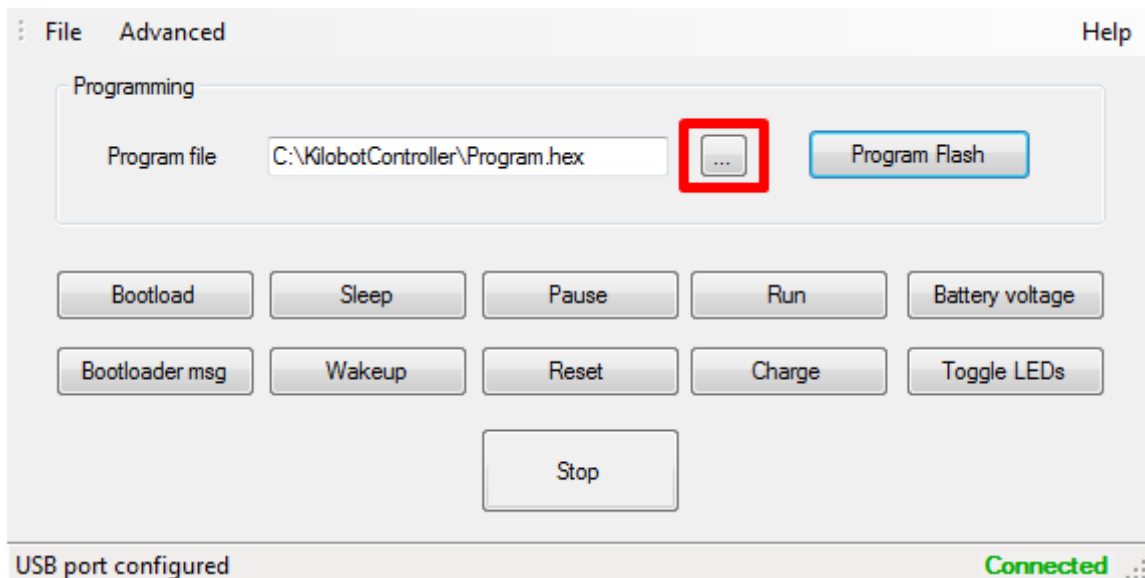


Figure 4.5.1 Kilobot Controller software

6. Press the “Program Flash” button to program the OHC (a black window will pop up).
7. Place *ALL* Kilobot robot in PAUSE mode (LED flashing yellow) underneath the OHC, and press the “Bootload” button in *KilobotController.exe*. The Kilobot will first quickly flash red->green->blue, to indicate it is now in the PROGRAMMING mode. Within 5 seconds, the LED will begin to flash blue and continue flashing while it is receiving the

new program. Once the LED has stopped flashing blue and switch to the flashing yellow of pause mode (approximately 30 seconds), press the “Stop” button in *KilobotController.exe*.

8. To run the program on the robot, press the “Run” button in *KilobotController.exe*. The new program should execute.
9. Press the “Pause” button in *KilobotController.exe* to stop the program. (Note: the robot needs to be in pause mode before programming can begin).

You can find in annex 5.1 an example code.

4.5.2 OHC Use

System Setup

Kilobots should be operated on a smooth, flat surface to ensure proper robot mobility. To aid communication, the surface should be glossy or reflective. A mirror or dry-erase whiteboard oriented horizontally is recommended.

The overhead controller should be hung above the Kilobots at a distance of about one meter. The robots beneath the OHC in a about a one meter diameter region will be able to receive communication from the OHC as shown in figure 4.5.2.

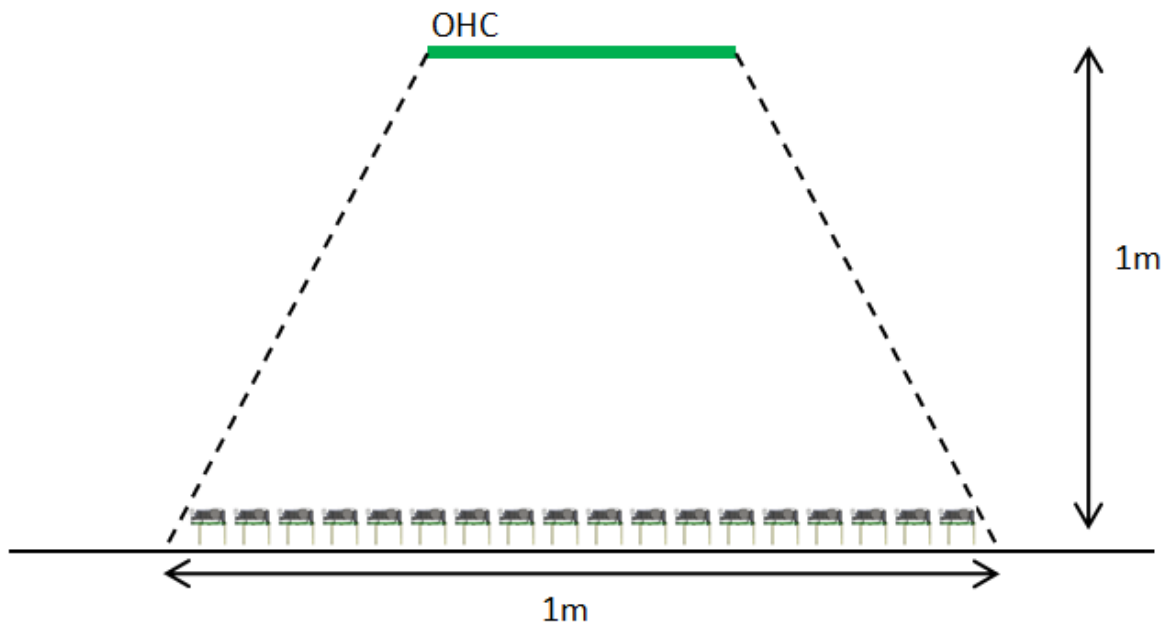


Figure 4.5.2 System setup

Overhead Controller (OHC) Interface Overview

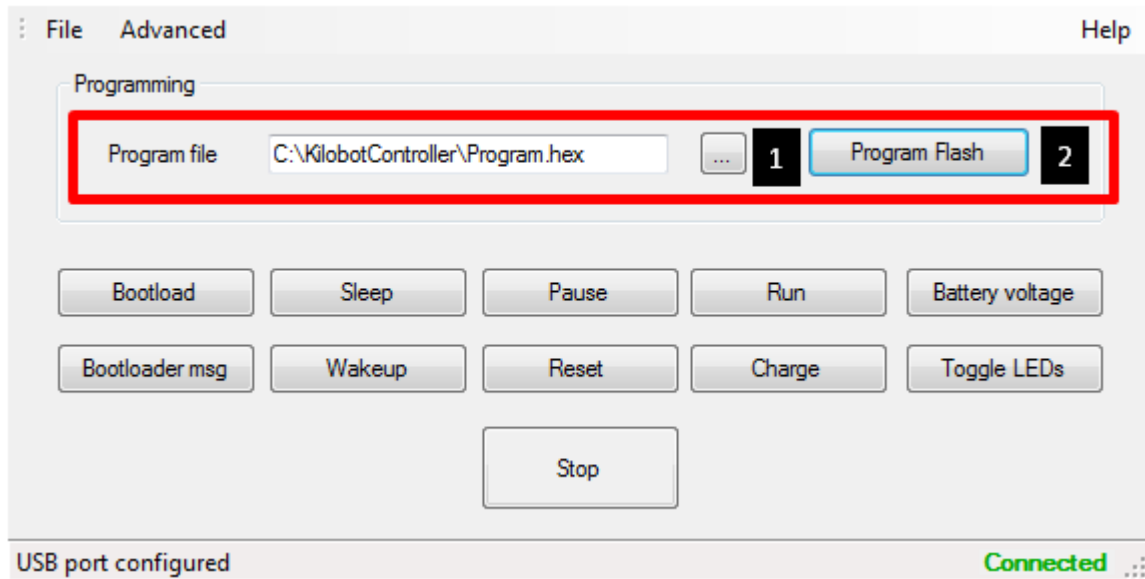


Figure 4.5.3 Programming Panel

1. Program file: browse for the compiled Kilobot user program [.hex] file to be sent to the Kilobots.
2. Program Flash: send the program and controller files to the OHC.

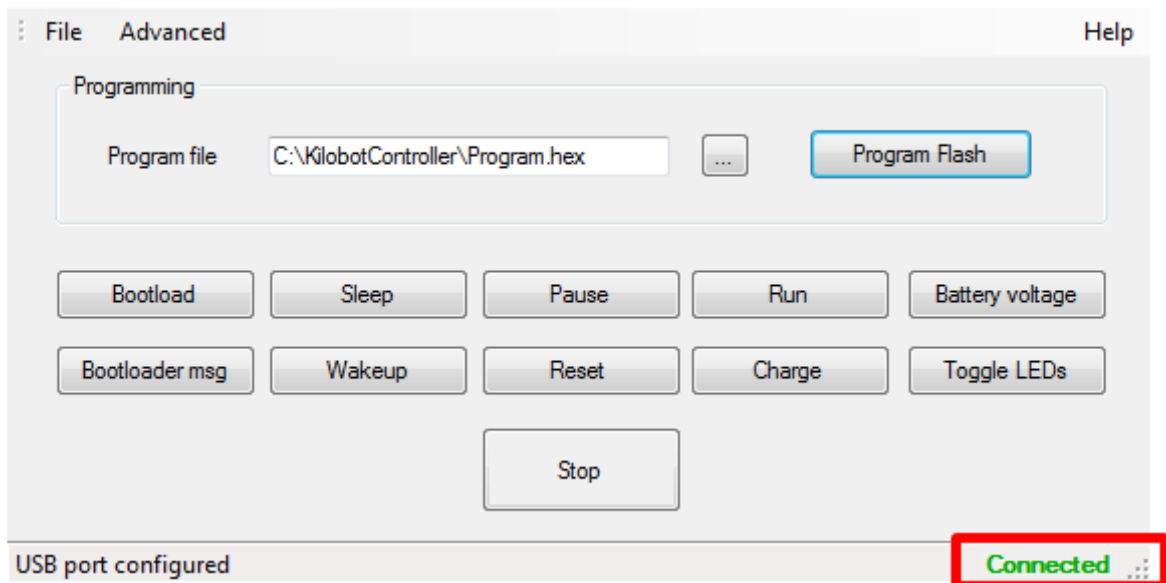


Figure 4.5.4 Connection Status

You can see the OHC is connected to the pc or not (fig 4.5.4).

Below in figure 4.5.5 is the Button Panel with a description of the buttons functionality.



Figure 4.5.5 Button Panel

1. **Bootload:** transmit the kilobot user program from the OHC to the Kilobots, which takes about 30 seconds. Ensure all robots are in pause mode before bootloading.
2. **Sleep:** put Kilobots in low-power SLEEP mode. They will wake up briefly every 60 seconds to check if a wakeup message is being sent. The robots conserve energy in this mode and can stay in sleep for months without recharging.
3. **Pause:** put Kilobots in idle PAUSE mode, and is also used to prepares robots to be reprogrammed.
4. **Run:** initiates Kilobots user program.
5. **Battery Voltage:** instructs Kilobots to indicate their battery voltage through the LED. where green indicates a battery voltage over 4V, blue indicates a voltage between 3.75 and 4, yellow indicates a voltage between 3.5 and 3.75, and red indicates a voltage under 3.5 volts .
6. **Bootloader msg:** sends a message for all robots to exit their program and jump to the bootloader.
7. **Wakeup:** take Kilobots out of low-power SLEEP mode. This message will transmit untill stop is pressed. It will take a minute or two for all the robots to wake up.
8. **Reset:** restarts the Kilobot program.
9. **Charge:** instructs Kilobots to enter CHARGE mode for battery recharging. The robots will stop the user program and blink red if they are actively charging, if not, the RGB led will turn off.
10. **Toggle LEDs:** flashes the LEDs on the OHC.
11. **Stop:** stops whatever the OHC is doing. This can be used to stop functions that are continuous such as wakeup or bootloading.

4.5.3 Kilobots API

The different functions specially developed for the Kilobot are explained below:

- **set_motor**(*cw_motor*,*ccw_motor*)
Set motor speed PWM values for motors between 0 (off) and 255 (full on) for *cw_motor* and *ccw_motor*
Example:
set_motor(100,100);
note that there are 4 calibration values used with the motors
cw_in_place - value for cw motor to turn the robot cw in place (note: ccw motor should be off)
ccw_in_place - value for ccw motor to turn the robot ccw in place (note: cw motor should be off)
cw_in_straight - value for the cw motor to move the robot in the forward direction
ccw_in_straight - value for the ccw motor to move the robot in the forward direction
- **_delay_ms**(*x*)
Busy wait for *x* milliseconds, interrupts can still trigger
//max value for milliseconds = 4000
Example:
_delay_ms(250);
- **set_color**(*r,g,b*)
Set LED color, values can be from 0(off)-3(brightest)
Example:
set_color(1,1,1);
- **kprinti**(*int*)
Print integer over serial port - be careful can effect timing!
Example:
kprinti(12);
- **kprints**(*string*)
Print string up to 10 characters - be careful, can effect timing!
Example:
kprints("HelloWorld");
- **message_out**(*tx0,tx1*)
Set message values to be sent over IR every .2 seconds, 2 bytes *tx0,tx1*
Example:
message_out(0x01,0xFF);

- **get_message()**
Take oldest message off of rx buffer message. It is only new if *message_rx[5]==1* !
If so, message is in *message_rx[0]* and *message_rx[1]*; distance to transmitting robot (in mm) is in *message_rx[3]*

Example:

```
get_message();  
if(message_rx[5]==1)  
{  
data=message_rx[0];  
distance=message_rx[3];  
}
```

- **enable_tx**
to turn off the transmitter, set `enable_tx = 0`
to turn on the transmitter, set `enable_tx = 1`

Example:

```
if(Want_To_Transmit==TRUE)  
enable_tx = 1;  
else  
enable_tx = 0;
```

- **measure_charge_status()**
Measure if battery is charging, returns 0 if no, 1 if yes

Example:

```
if(measure_charge_status())  
charging = TRUE;  
else  
charging = FALSE;
```

- **measure_voltage()**
Measure battery voltage, returns voltage in .01 volt units
for example if 394 is returned, then the voltage is 3.94 volts

Example:

```
current_voltage = measure_voltage();
```

- **get_ambient_light()**
Returns the value of ambient light
note: will return -1 if there is an incoming message (which also uses a/d)
note: turns off interrupts for a short while to sample a/d

Example:

```
local_brightness = get_ambient_light();
```

Numerous other functions and AVR-C programming examples can be found on the web.

5. Annexes

5.1 Examples code

These examples of code have different purposes described in there titles.

5.1.1 Transmits data to neighbors, and blinks led when the message is received:

```

...
message_out(255,255);//set message to be sent over IR
enable_tx=1;//enable transmission of message every .2 seconds

//check for message
get_message();
if(message_rx[5]==1)//new message has been received
{
    set_color(1,1,1);//turn RGB LED white
    kprinti(message_rx[0]);//send first byte of received message over serial debug
cable
    kprinti(message_rx[1]);//send second byte of received message over serial debug
cable
    kprinti(message_rx[3]);//send measured distance from transmitting robot over
serial debug cable
    kprints("      ");
    _delay_ms(10);//wait 10 ms
    set_color(0,0,0);//turn RGB LED off
}

```

5.1.2 Non-blocking timed movement:

```

...
static int mode=0;//mode represents state
if(clock<40000)
{
    //clockwise rotation
    if(mode!=0)

```



```

        {
            set_motor(0,0xa0);//spin up cw motor to overcome friction
            _delay_ms(30);
            set_motor(0,cw_in_place);//set cw motor to calibrated value for good cw
rotation
            mode=0;
        }

    }
    else if(clock<80000)
    {
        if(mode!=1)
        {
            //counter clockwise rotation
            set_motor(0xa0,0);//spin up ccw motor to overcome friction
            _delay_ms(30);
            set_motor(ccw_in_place,0);//set ccw motor to calibrated value for good cw
rotation
            mode=1;
        }
    }
    else if(clock<120000)
    {
        if(mode!=2)
        {
            //move forward
            set_motor(0xa0,0xa0);//spin up both motors to overcome friction
            _delay_ms(30);
            set_motor(ccw_in_straight,cw_in_straight);//set both motor to calibrated
value for good forward movement
            mode=2;
        }
    }
    else if(clock<160000)
    {
        if(mode!=3)
        {
            set_motor(0x00,0x00);//turn off motors
            mode=3;
        }
    }
    else if(clock<200000)
    {
        mode=5;
        clock=0;//reset clock to start cycle over agian
    }
}

```

5.1.3 Random move in see neighbors, stop else

```

static int one_time=0;
if(one_time==0)//if this is the start of the program, only run this if statement once
{
    for(int i=0;i<30;i++)
        randseed+=measure_voltage( );//generate some random sensor data
    one_time=1;//indicate that the first time if statement has now run
    srand(randseed);//seed random variable with some sensor data
}
static int neighbor=0;//1 if i've seen any neighbors, 0 if not
static int movement=0;//1 if moving, 0 if stopped
static int start_move=0;//1 if the movement has changed, 0 if not

//check to see if i have received any new messages, if so, then i have seen neighbors
get_message();
if(message_rx[5]==1)
    neighbor=1;

//if i am currently not moving, and want to start moving
if(start_move==1)
{
    if(movement==0)
    {
        set_motor(0xa0,0xa0);//spin up motors
        _delay_ms(30);
        set_motor(ccw_in_straight,cw_in_straight);//set to move straight
    }
    else if(movement==1)
    {
        set_motor(0xa0,0);//spin up motor
        _delay_ms(30);
        set_motor(ccw_in_place,0);//set to move ccw
    }
    else
    {
        set_motor(0,0xa0);//spin up motor
        _delay_ms(30);
        set_motor(0,cw_in_place);//set to move cw
    }
    start_move=2;//mark that i am currently moving
}
else if(start_move==0)//i do not want to move
{
    set_motor(0,0); //stop motors
}
//every 2000 counts of clock, check to see if i've seen any neighbors
//if i have, there is a 20% chance i will change my movement (such as straight to ccw)
if(clock>2000)

```

```

{
  if((rand()%100)<20)//update robot movement 20% of the time
  {
    int rand_d=rand()%100;
    if(rand_d<25)//if updating robot movement 25% of the time move ccw
    {
      if(movement!=1)
      {
        movement=1;
        if(start_move==2)
          start_move=1;
      }
    }
    else if(rand_d<50)//if updating robot movement 25% of the time move cw
    {
      if(movement!=2)
      {
        movement=2;
        if(start_move==2)
          start_move=1;
      }
    }
    else//if updating robot movement 50% of the time move straight
    {
      if(movement!=0)
      {
        movement=0;
        if(start_move==2)
          start_move=1;
      }
    }
  }
  //if i've received a message from a neighbor in the last 2000 counts of clock
  //and i am not moving, start moving, and turn my rgb led red
  if(neighbor==1)
  {
    set_color(1,0,0);
    if(start_move==0)
      start_move=1;
  }
  else
  {
    //if i have not received a message from a neighbor in the last 2000 counts of clock
    //stop moving, and turn my RGB led off
    set_color(0,0,0);
    start_move=0;
  }
}

```

```
clock=0;//reset clock  
neighbor=0;//reset neighbor count
```

```
}
```

6. WARRANTY

K-TEAM warrants that the Korebot II is free from defects in materials and workmanship and in conformity with the respective specifications of the product for the minimal legal duration, respectively one year from the date of delivery.

Upon discovery of a defect in materials, workmanship or failure to meet the specifications in the Product during the afore mentioned period, Customer must request help on K-Team Internet forum on <http://www.k-team.com/forum/> by detailing:

- the type of Korebot II used (version)
- the kernel version of the Korebot II
- the programming environment of the Korebot/robot (standard, version, OS)
- the standard use of Product before the appearance of the problem
- the description of the problem.

If no answers have been received within two working days, Customer can contact K-TEAM support by phone or by electronic mail with the full reference of its order and Korebot II serial number.

K-TEAM shall then, at K-TEAM's sole discretion, either repair such Product or replace it with the equivalent product without charging any technical labor fee and repair parts cost to Customer, on the condition that Customer brings such Product to K-TEAM within the period mentioned before. In case of repair or replacement, K-TEAM may own all the parts removed from the defective Product. K-TEAM may use new and/or reconditioned parts made by various manufacturers in performing warranty repairs and replacement of the Product. Even if K-TEAM repairs or replaces the Product, its original warranty term is not extended.

This limited warranty is invalid if the factory-applied serial number has been altered or removed from the Product.

This limited warranty covers only the hardware and software components contained in the Product. It does not cover technical assistance for hardware or software usage and it does not cover any software products contained in the Product. K-TEAM excludes all warranties expressed or implied in respect of any additional software provided with Product and any such software is provided "AS IS" unless expressly provided for in any enclosed software limited warranty. Please refer to the End User License Agreements included with the Product for your rights with regard to the licensor or supplier of the software parts of the Product and the parties' respective obligations with respect to the software.

This limited warranty is non-transferable.

It is likely that the contents of Customer's flash memory will be lost or reformatted in the course of the service and K-TEAM will not be responsible for any damage to or loss of any programs, data or other information stored on any media or any part of the Product serviced hereunder or damage or loss arising from the Product not being available for use before, during or after the period of service provided or any indirect or consequential damages resulting therefore.

IF DURING THE REPAIR OF THE PRODUCT THE CONTENTS OF THE FLASH MEMORY ARE ALTERED, DELETED, OR IN ANY WAY MODIFIED, K-TEAM IS NOT RESPONSIBLE WHATEVER. CUSTOMER'S PRODUCT WILL BE RETURNED TO CUSTOMER CONFIGURED AS ORIGINALLY PURCHASED (SUBJECT TO AVAILABILITY OF SOFTWARE).

Be sure to remove all third parties' hardware, software, features, parts, options, alterations, and attachments not warranted by K-TEAM prior to Product service. K-TEAM is not responsible for any loss or damage to these items.

This warranty is limited as set out herein and does not cover, any consumable items (such as batteries) supplied with the Product; any accessory products which is not contained in the Product; cosmetic damages; damage or loss to any software programs, data, or removable storage media; or damage due to (1) acts of God, accident, misuse, abuse, negligence, commercial use or modifications of the Product; (2) improper operation or maintenance of the Product; (3) connection to improper voltage supply; or (4) attempted repair by any party other than a K-TEAM authorized module service facility.

This limited warranty does not apply when the malfunction results from the use of the Product in conjunction with any accessories, products or ancillary or peripheral equipment, or where it is determined by K-Team that there is no fault with the Product itself.

K-TEAM EXPRESSLY DISCLAIMS ALL OTHER WARRANTIES THAN STATED HEREINBEFORE, EXPRESSED OR IMPLIED, INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE TO THE FULLEST EXTENT PERMITTED BY LAW.

Limitation of Liability: IN NO EVENT SHALL EITHER PARTY BE LIABLE TO THE OTHER FOR ANY INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES RESULTING FROM PERFORMANCE OR FAILURE TO PERFORM UNDER THE CONTRACT, OR FROM THE FURNISHING, PERFORMANCE OR USE OF ANY GOODS OR SERVICE SOLD OR PROVIDED PURSUANT HERETO, WHETHER DUE TO A BREACH OF CONTRACT, BREACH OF WARRANTY, NEGLIGENCE, OR OTHERWISE. SAVE THAT NOTHING HEREIN SHALL LIMIT EITHER PARTY'S LIABILITY FOR DEATH OR PERSONAL INJURY ARISING FROM ITS NEGLIGENCE, NEITHER PARTY SHALL HAVE ANY LIABILITY TO THE OTHER FOR INDIRECT OR PUNITIVE DAMAGES OR FOR ANY CLAIM BY ANY THIRD PARTY EXCEPT AS EXPRESSLY PROVIDED HEREIN.



K-Team S.A.
RUE GALILEE 9
1400 YVERDON-LES-BAINS
SWITZERLAND
