# KILOBOT

user manual


VERSION 2.0
JANUARY 2016

**Documentation Author**

Julien Tharin
K-Team S.A.
Z.I des Plans-Praz
CH-1337 Vallorbe
Switzerland

Email: info@k-team.com
Url: www.k-team.com

**Documentation version**

| Version | Date | Author | Description |
|---|---|---|---|
| 2.0 | 15.01.2016 | F.Lambercy | First draft |
| | | | |
| | | | |
| | | | |
| | | | |

## Trademark Acknowledgements:

| | | |
|---|---|---|
| **Kilobot:** | **:** | Harvard University |
| **Khepera** | **:** | K-Team SA |
| **V-REP** | **:** | Coppelia Robotics |

## 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

Thank you for buying Kilobot robots!

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.

You can find more information directly on the designer website:
http://www.eecs.harvard.edu/ssr/projects/progSA/kilobot.html

This version of manual explain how to use the new version of Kilobot firmware developed by Harvard. More information on this version can be found on the website:
https://www.kilobotics.com/

If you already own Kilobot robot and OHC with older firmware, please consult the guide ***kilobot_update.doc*** to update your older system.

## 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.
- **Keep the board away of conducting materials**. To avoid any short-circuit and damage to the robots, avoid contact with metallic lonely wires and other conducting materials.

## 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 328p (8bit @ 8MHz)
- Memory :
  - 32 KB Flash
  - 1KB EEPROM
- Battery/ autonomy: Rechargeable Li-Ion 3.7V / 3-10 hours continuously, 3 months in sleep mode
- Charging : Kilobot charger for 10 robots simultaneously (optional)
- Communication : IR (up to 7cm, up to 32kb/s and 1kbyte/s with 25 robots), serial (256000 baud)
- Sensing : 1 IR and 1 light intensity
- Movement : forward, rotation (1cm/s , 45deg/s)
- Light : one RGB led
- Software : Kilobot Controller software for controlling the robot
- Programming : C language with WinAVR compiler combined with Eclipse or the online Kilobotics editor
- Dimensions : diameter: 33 mm, height 34 mm (including the legs, without recharge antenna).

# 2. UNPACKING AND INSPECTION

## 2.1 Package Contents

They are 3 different packages sold separately. They are containing the following items:

- The robot package:

  1 CD-ROM*

  10 Kilobot robots

- The overhead controller package:

  1 overhead controller

  1 CD-ROM*

  3 cables (1 usb, a 6-wire cable and a 2–wire cable)

- The charger package:

  1 charger

  1 transformer

  1 CD ROM*

* If multiple packages are ordered together, only one CD-ROM will be shipped.

## 2.2 Inspection

The material should be checked after unpacking.

For the Kilobots robots, check that their legs are not bended. The charging tab also must not be bended. The jumper and the battery should be present (see figure 3.1). The battery should be plugged with the negative on top (opposite of the battery holder +/- marks).

For the Overhead controller, check that the LEDS are not bended or broken. The jumper for firmware programming should be present (see figure 3.2).

# 3. DESCRIPTION

## 3.1 Overview

### 3.1.1 Kilobot robot

An overview of the Kilobot hardware is depicted in the Figure 3.1. The locations of various key elements are indicated for later references.

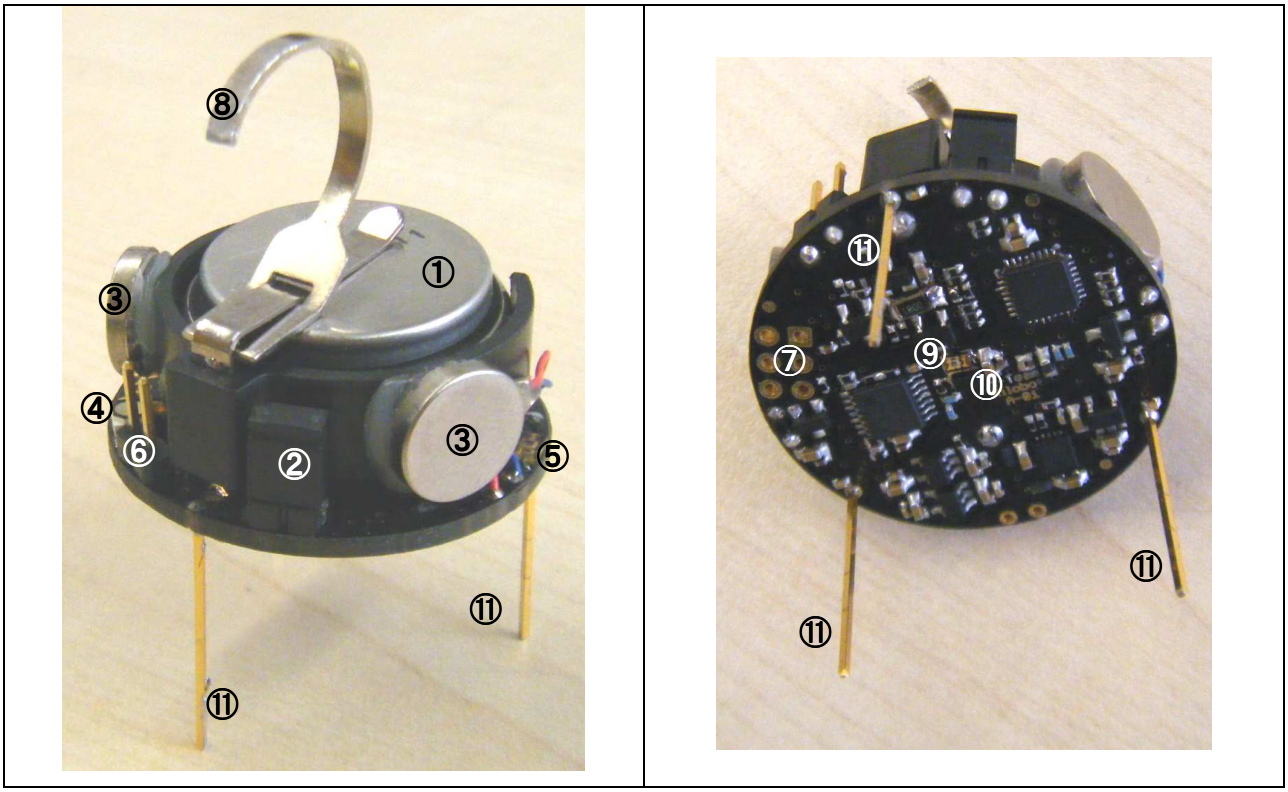



***Figure 3.1: Kilobot overview***

① 3.7-Volt Battery (- up, + down)

② Power jumper

③ Vibration motors

④ LED (Red/Green/Blue)

⑤ Ambient light sensor

⑥ Serial output header

⑦ Direct programming socket

⑧ Charging Tab

⑨ IR Transmitter

⑩ IR Receiver

⑪ Robot leg

### 3.1.2 Overhead controller (OHC)

An overview of the Kilobot Overhead controller (OHC) is depicted in the Figure 3.1. The locations of various key elements are indicated for later references.

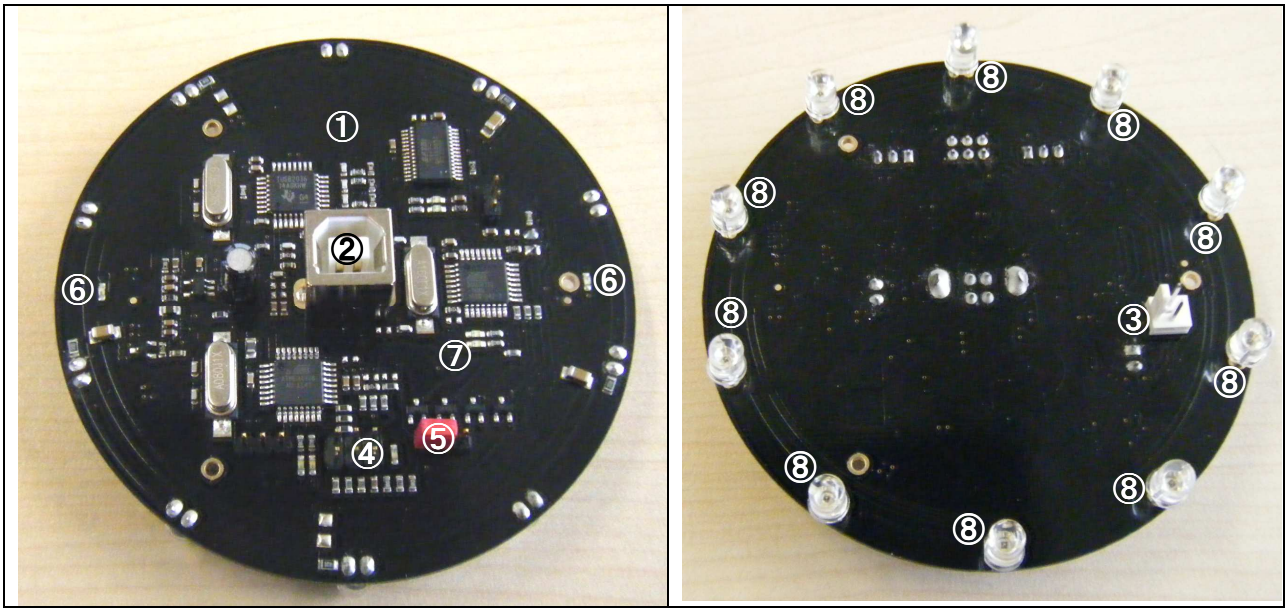



***Figure 3.2: Overhead controller overview***

① Overhead controller board

② Connector for USB cable

③ Connector for debug cable

④ Connector for firmware programming

⑤ Firmware programming jumper

⑥ Diode for OHC connection test

⑦ Power on LED

⑧ IR LED

## 3.2 Kilobot Hardware

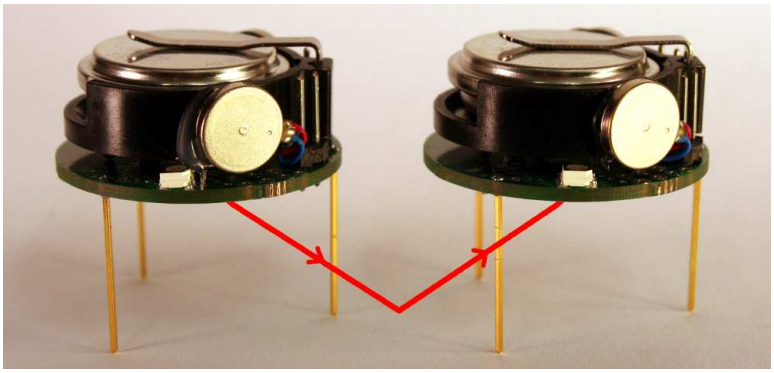
The hardware of the **Kilobot** is described thereafter.

The robot has a microcontroller ATmega 328p (8bit @ 8MHz) as CPU. It contains these following types of 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.

Its other features are described in detail below:

- Battery : Rechargeable Li-Ion 3.7V, for a 3 months autonomy in sleep mode. Each Kilobot has a built-in charger circuit, 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 distance depends on the surface used as the light intensity is used to compute the value.
  - 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 (kiloGUI) 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 Eclipse 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.

# 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 and an USB port (not included)
- Kilobot robot
- Over-head controller (OHC)
- Kilobot charger

### 4.1.2 Required software :

To start programming the Kilobot with the new version from kilobotics, you have two solutions. You can either use the online editor and compile without the need of any installation (https://www.kilobotics.com/editor ).

Or you can install WinAVR and Eclipse to compile the whole library on your computer (https://github.com/mgauci/kilobot_notes/blob/master/eclipse_winavr_setup/eclipse_winavr_setup.md ).

Required files:

- WinAVR
  http://winavr.sourceforge.net/
- Eclipse IDE for C/C++ Developers:
  http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/lunasr2

These files are included in the delivered from the CD_ROM:

- Kilobot programming project files (VERSION is the version of the files):
  ***software\source_code\Kilobot VERSION.zip***
  This archive contains:
  - ***Eclipse/*** development directory for Eclipse
  - ***Firmware/*** compiled firmware to update your system or test actual demo program
  - ***KiloGUI/*** Kilobot Controller program
  - ***Kilolib-master/*** Source code of the Kilobot Library
  - ***Kilobot_update.doc*** documentation to upgrade your old Kilobot system to this new version
  - ***Zadig_2.1.2.exe*** Software to modify the USB driver to work with new controller
  - ***Firmware/bootloader.hex*** Bootloader of the Kilobot. Useful if an upgrade is needed or if the Kilobot firmware is corrupted.
  - ***Firmware/controller.hex*** Firmware of the OHC controller.
  - ***Firmware/Kilobot2.0.hex*** Factory default firmware of the Kilobot
  - ***Firmware/orbit.hex*** Kilobot demo program
  - ***Firmware/orbit-star.hex*** Kilobot demo program
  - ***Firmware/simplemov.hex*** Kilobot demo program

## 4.2 Installation

The following sub-chapters explain the software and installations for local compiling. These explanations are from the website below. You will find more details here: https://github.com/mgauci/kilobot_notes/blob/master/eclipse_winavr_setup/eclipse_winavr_setup.md

### 4.2.1 Install the software

1. Install WINAVR compiler if not already done.
2. Copy all the files from ***software\source_code\Kilobot VERSION/Eclipse*** on your computer. In this directory, the kilolib is already compiled by K-Team (renamed ***libkilolib.a***).
3. Install also Eclipse IDE for C/C++ Developers: http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/lunasr2
4. Setup Eclipse. As the source file were already compiled under Eclipse, you just need to select your ***software\source_code\Kilobot VERSION/Eclipse*** as workspace directory. Once done, try to compile the source code to validate the installation.
5. You can also install AVRstudio if you need to update your Kilobot System or if you want to restore the bootloader

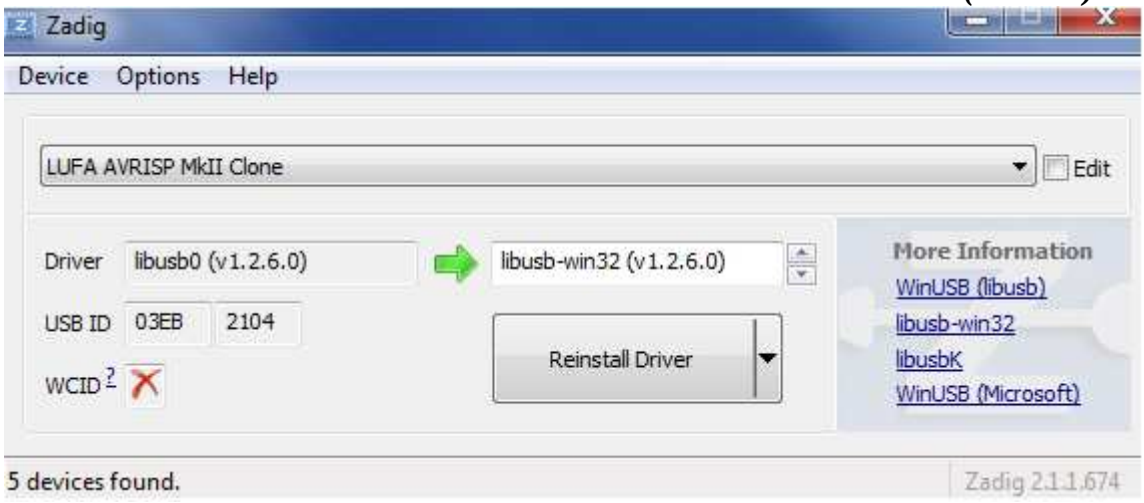
### 4.2.2 Overhead Controller (OHC) Software Installation

The OHC software did need any installation. Simply run the KiloGUI to have access to your OHC.
For other platform than Windows, please visit the kilobotics website (https://www.kilobotics.com/download )

#### 4.2.2.1 Updating driver of the OHC

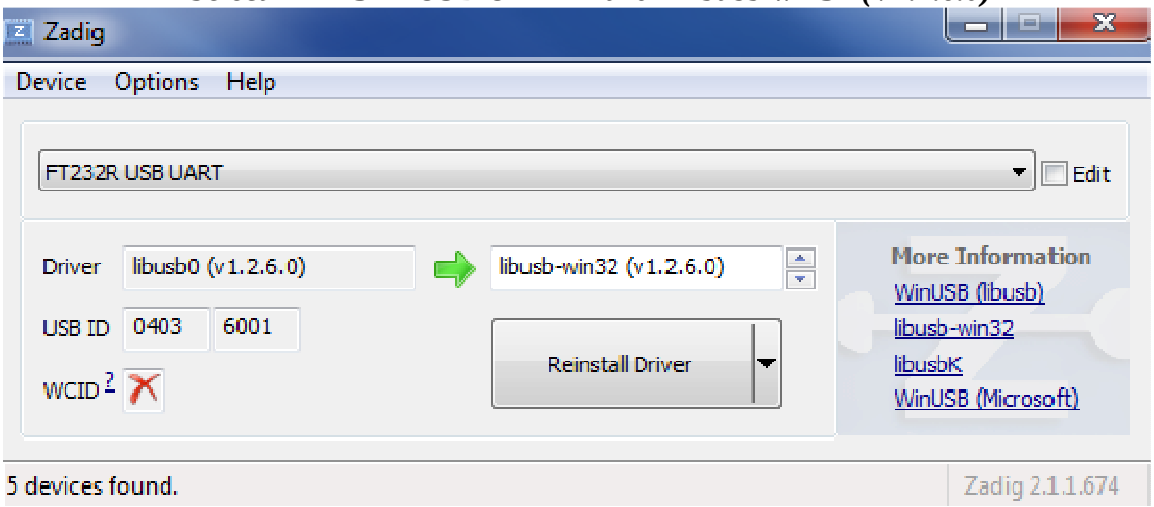
If your OHC is not recognize by the KiloGUI interface, you will need to modify the driver:

- Execute the software « ***zadig_2.1.1.exe*** »
- Option -> list all device
- Select « ***LUFA AVRISP MkII Clone*** » and « ***libusb-win32 (V1.2.6.0)*** »




- Click « ***Replace Drive*** » (or "***Reinstall Driver***" if already done once).

- Select « ***FT232R USB UART*** » and « ***libusb-win32 (V1.2.6.0)*** »




- Click « ***Replace Driver*** ».
- Close « Zadig » interface

Now the KiloGUI interface must recognize your OHC

### 4.2.3 Overhead Controller (OHC) Hardware Installation

1. Check that the firmware jumper is plugged at the correct place (figure 4.6).
2. Connect the USB cable to the PC; the power LED of the OHC should turn green.
3. Run the "***kilogui.exe***" program, the message ***"connected"*** must appears at the bottom of the windows:


4. Press the ***"LedToggle"*** button to validate the communication (a Green LED on the OHC must change its state at each pressing).
5. If the message ***"connected"*** didn't appear, please follow the steps in chapter 4.2.2.1

### 4.2.4 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. In AVR Studio, select Tools->ProgramAVR->Autoconnect .

2. In the window that pops up, under the program tab, in the flash category select the input hex file to be ***"bootloader.hex".***

3. Put the firmware programming jumper of the OHC in the "firmware mode" position and connect the firmware cable (figure 4.3).

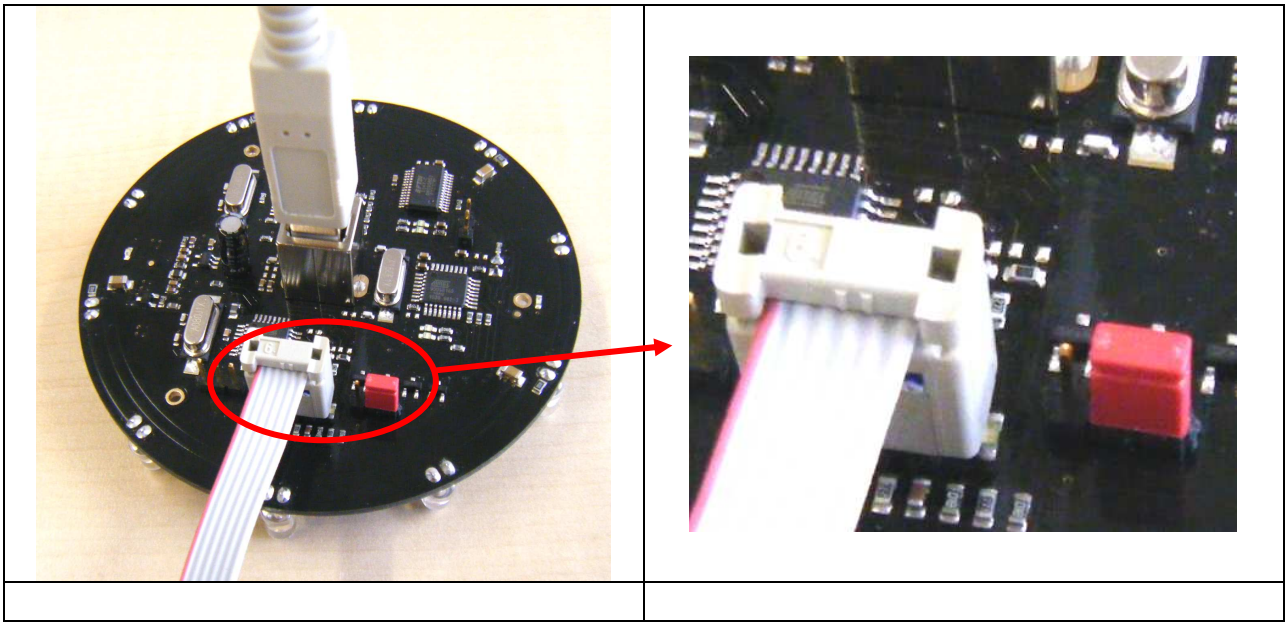


***Figure 4.3:firmware cable and jumper in firmware mode***

4. Turn on the robot by adding the power jumper as shown in figure 4.4.

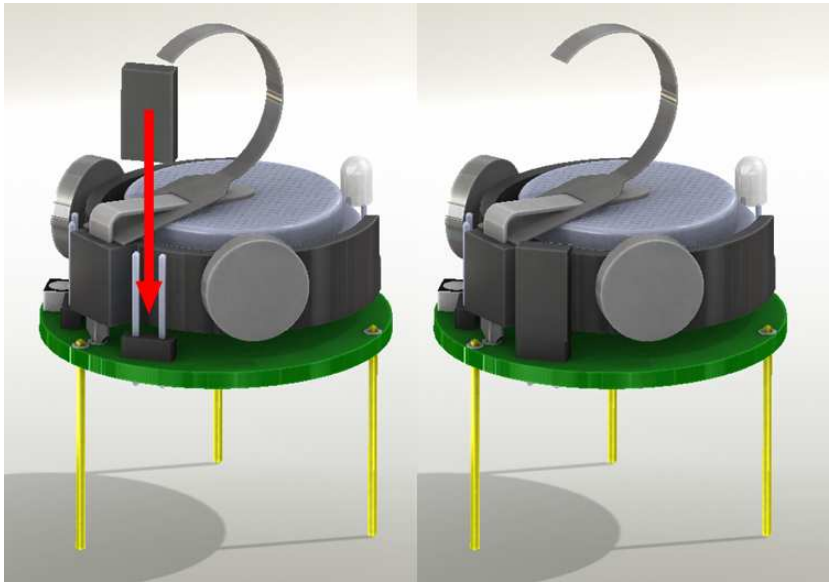


***Figure 4.4: power jumper***

5. Connect the AVRisp programmer to the robot as shown in figure 4.5. 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.

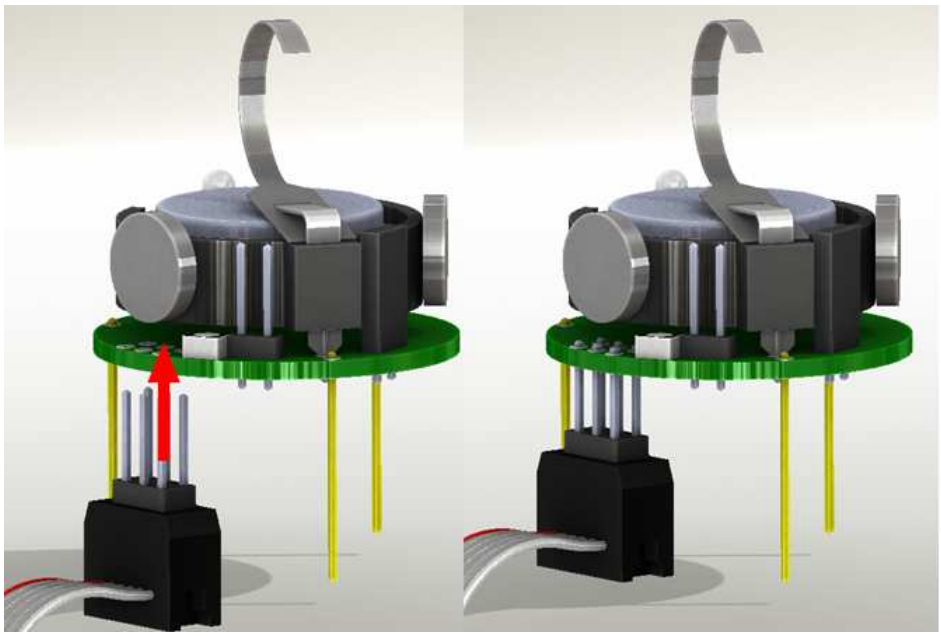


***Figure 4.5: communication connector***

6. 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

7. 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.

8. Put the firmware programming jumper of the OHC in the back to the standard position.

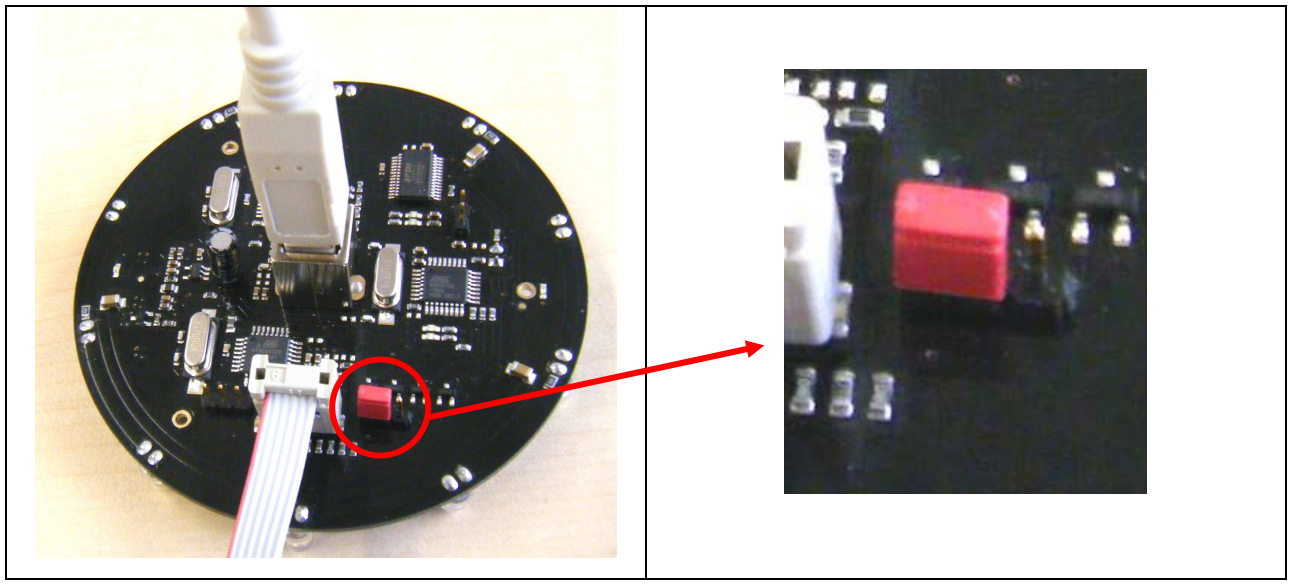


***Figure 4.6: firmware jumper: normal mode***

## 4.3 Usage

### 4.3.1 Programming online (kilobotics.com)

An easy way to program the robots is to use the Web-based Editor. To start, you need to have an account with Dropbox and you need to install the Dropbox software on your computer. The first time you open the Editor, will be asked to sign into Dropbox. Then use the editor to create a new file by clicking on +New. This will create an example file `untitled.c` with the basic loop and setup definitions in place. Rename this file to something you like, and then use the edit icon on the right hand side to open the file for editing. You can also now compile the file, by clicking on the green Compile button. This will produce a `yourfilename.hex` file. Both code and compiled files are stored in `Dropbox/Apps/KiloEdit`. Now use the KiloGUI to upload the hex file to your kilobots.

Here are some resources on writing programs

1. The manual for C programming language and the AVR specific functions (TBA)
2. Kilobot Library API: All the robot functions and their usage (http://www.kilobotics.com/docs ).
3. Labs: A tutorial will step by step example programs; great for a class or to get new students started (http://www.kilobotics.com/labs ).

Note that you do not need to use the editor to modify your programs; you can open your code in any editor of choice (like emacs or vi) and then use the KiloEditor just for compiling. You can also bypass the editor all together by installing the AVR-C compiler directly on your machine; as explain in chapter 4.3.2.

The default factory firmware can be found on the CD (“***software\source_code\Kilobot VERSION/Online Source Code/Kilobot2_test.c”***). You can add this file to your dropbox directory if you want to modify it. Another demo program can be found on the Kilobotics website (https://github.com/SSR-Harvard/kilobotics-labs ).

### 4.3.2 Programming with Eclipse

Open Eclipse C/C++ editor and select the ***software\source_code\Kilobot VERSION/Eclipse*** directory as workspace for your environment.

The initial source code (***main.c***) is an example where the Robot move alternatively on the left/right/forward if no other Kilobot is in sight. Once at least another Kilobot is detected, the robot will stop and indicated with the RGB LED the distance of the other Robot.
This is the default factory demo code.

If you want to keep the initial behavior and add your code, find the text below to see where you need to add your code:

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

Once you own code added, you can compile the source code (***Project->Build All***). If no error is found, you will find your new compiled firmware in ***Eclipse/Kilobot2.0/Release/Kilobot2.0.hex***.

You will need to upload this file to your Kilobot as explain in chapter 4.3.3.

### 4.3.3 Uploading code in a group of Kilobot

1. Run ***kilogui.exe***, click the button ***[select a file]*** to browse your files and select the firmware to upload (**!!!be careful to not select an OHC firmware, this can corrupted the Kilobot bootloader or even damage the robot!!!)**.

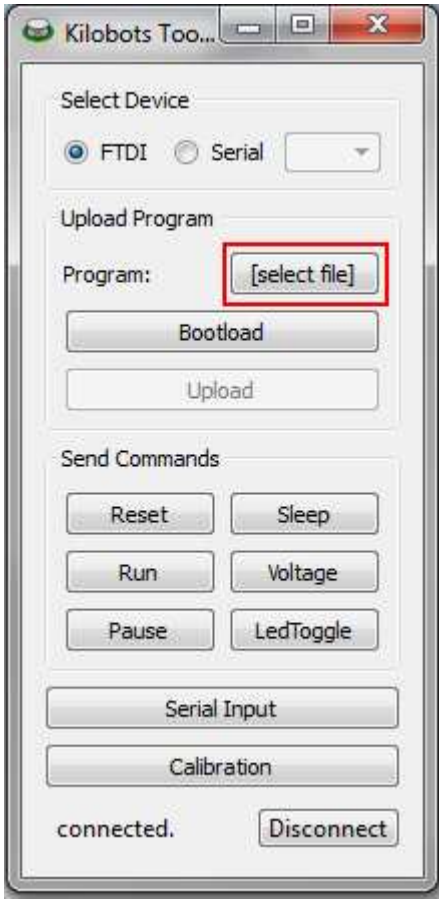



***Figure 4.7 : Kilobot Controller software***

2. Place *ALL* Kilobot robot in PAUSE mode (LED flashing green) underneath the OHC, and press the "***Bootload***" button in ***Kilogui.exe***. The Kilobot will turn their LED blue to indicate that they are ready to be programmed.
3. Then press the ***"Upload"*** button to start programming. The Kilobot will flash Green and Blue alternatively.
4. Once the Kilobot programmed, they will return to PAUSE state (LED flashing green).
5. To run the program on the robot, press the "Run" button in *KiloGUI.* The new program should execute.
6. Press the "Pause" button in *KiloGUI* to stop the program. (Note: the robot needs to be in pause mode before programming can begin).

*Note: once one button is pressed on KiloGUI, the action will be repeat infinitely. To avoid any trouble with IR communication, press again the same button to stop the repeat.*

You can find in annex 5.1 examples of source code.

### 4.3.4 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 4.8.

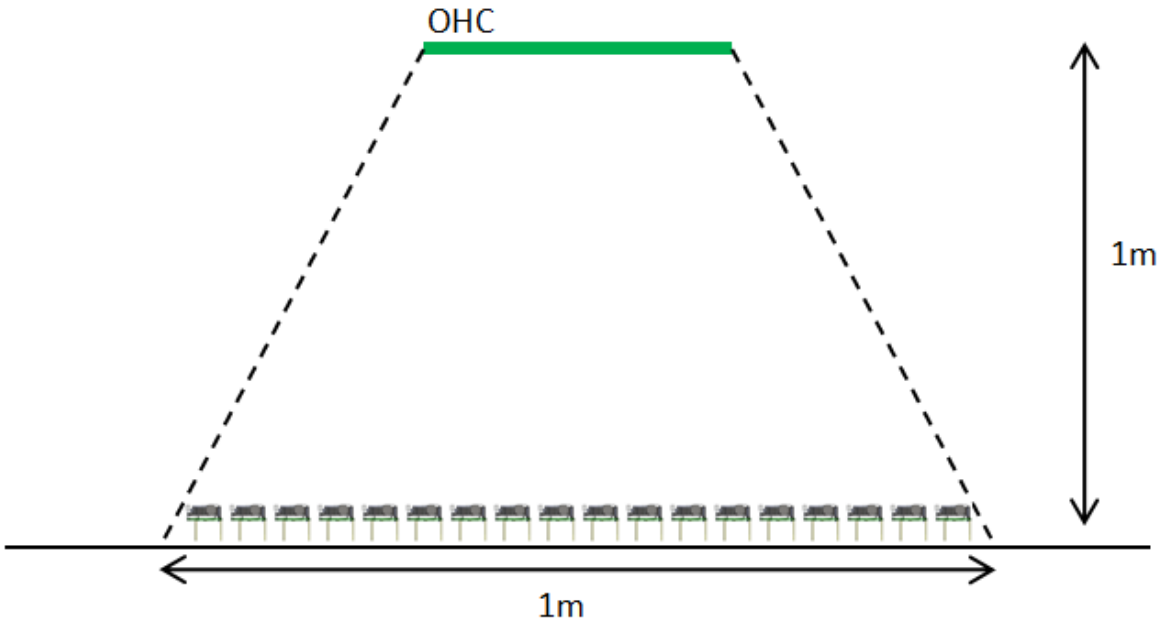



***Figure 4.8: System setup***

**Overhead Controller (OHC) Interface Overview**

1. **Select File:** browse your files to select the firmware to upload.

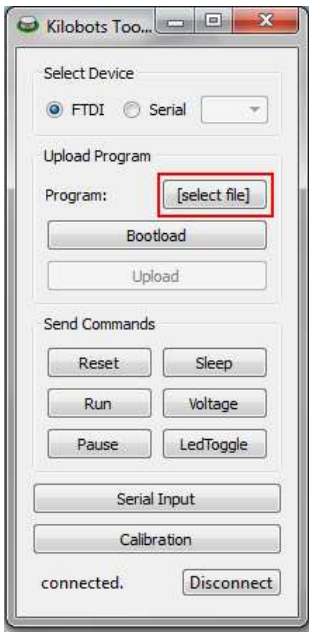



***Figure 4.9: browse your files***

2. **Bootload**: place the Kilobot in programing mode

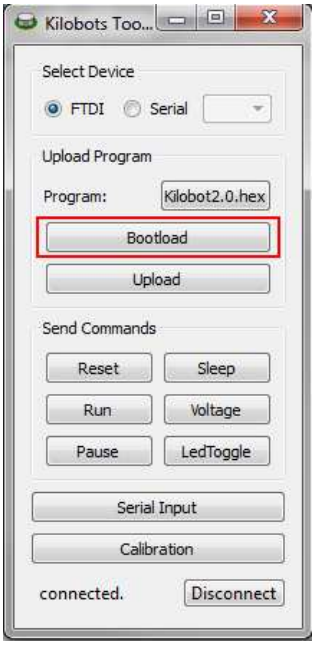



***Figure 4.10: Connection Status***

3 **Upload**: start programming the Kilobot


***Figure 4.11: upload button***

4 **Command panel**:

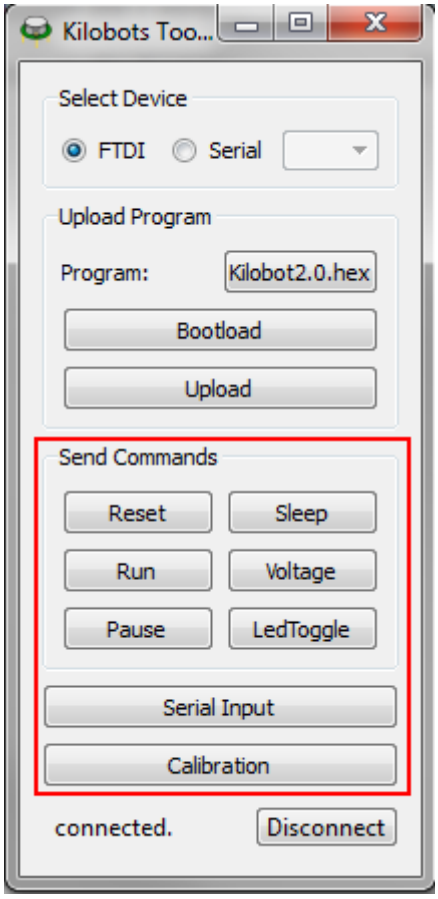



***Figure 4.12 : Send commands panel***

1. **Reset:** Jump to the user program starting point, resetting all state, and stop. Remain idle (blinking green) waiting for the next command.
2. **Run:** Run the user program.
3. **Pause**: Pause the user program (preserves state, so that when you click run the program resumes where it left off). Use Reset if you want to stop the robots and start the program from the beginning.
4. **Sleep**: Switch to low-power sleep mode (leds will flash white once every 8 seconds).
5. **Voltage**: Display voltage level using LED (blue/green = fully charged, yellow/red = need battery to be recharged)
6. **LedToggle**: Toggle LEDs on controller, used to check communication between PC and controller.
7. **Serial Input**: Display received messages from Kilobot using 2-wire serial cable
8. **Calibration**: Set the UID of a Kilobot and their motor values. See below for more details

### 4.3.5 Charger

A charger for up to 10 Kilobots is available as an optional item.

1. Put the Kilobots charger on a flat surface, having the bar with the red indicator below and the black above.

   WARNING: if the color position is not respected, that will not work and may damage the charger or the robots.

2. Plug the Kilobots charger transformer to the charger and to a power socket. Pay attention to the polarization (+ outside, - inside).

   WARNING: if the polarization is not respected, that will not work and may damage the charger or the robots.

3. Switch the Kilobots on (put the power jumper as in figure 4.4).

4. Pay attention to respect the bar side and the cable polarization as explained above in 1) and 2), Then hang the Kilobots as specified in the figure below. The bar with the red indicator is in contacts of the legs of the robots and is the positive.

***Note: in this new version of software, there's no more Charge status on the Kilobot. To improve the charging time, place the Kilobot in Sleep mode. You can check the battery status with the Voltage button on KiloGUI. Normally, a complete charge take approximately 4 hours.***

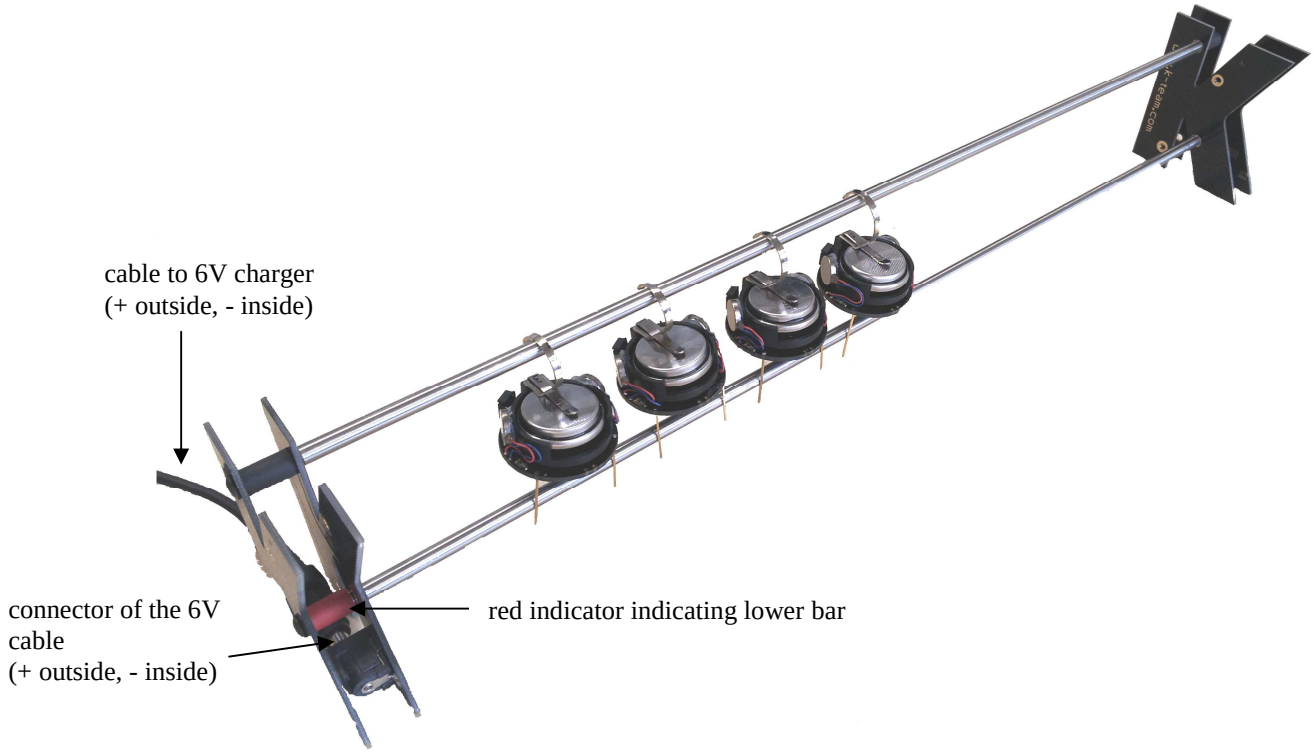



***Figure 4.13: Kilobots on charger***

### 4.3.6 Debugging

With the ***printf()*** function, you can get a feedback from the robot. You must connect the robot to the OHC with the 2-wire cable, respecting the polarity on the robot as shown in the figure below (see figures 3.1 and 3.2 for the connectors place).

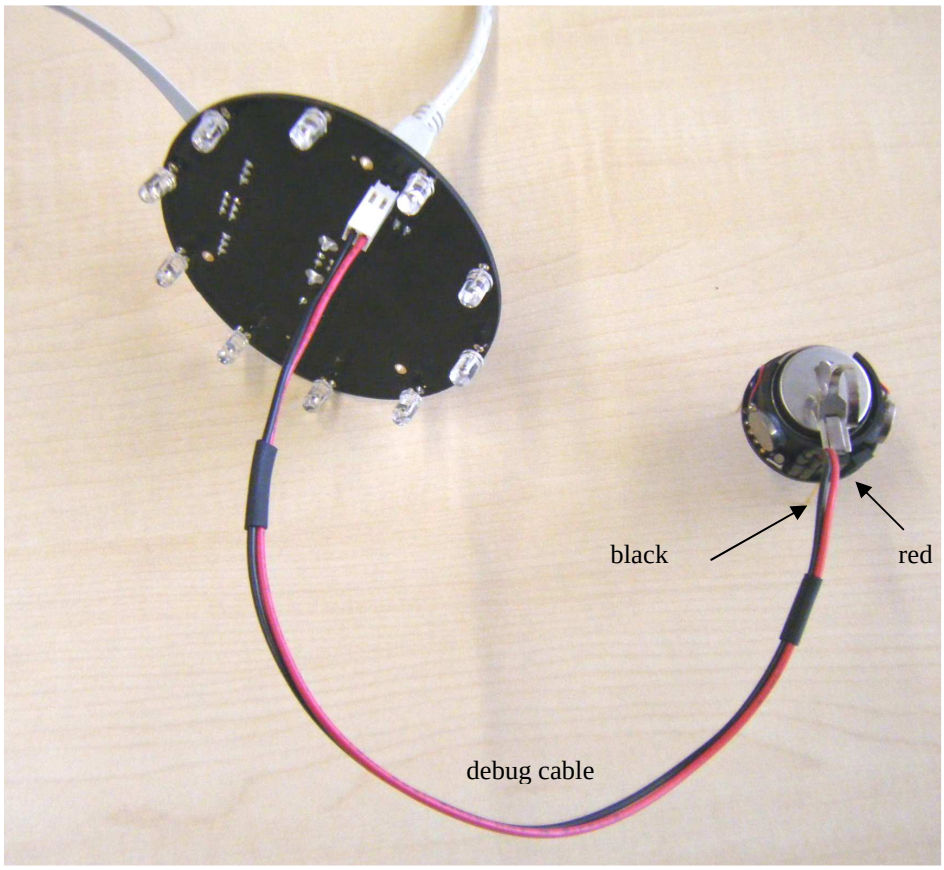



Figure 4.14 Debug settings

1. Open KiloGUI interface
2. Launch your program with the “Run” button of the KiloGUI.
3. Press the serial input button to open a pop-up where all the message sent by the Robot will appear.

### 4.3.7 Kilobots API

The different functions specially developed for the Kilobots are explained below. Numerous other functions and AVR-C programming examples can be found on the web.

You can also found more documentation on the Kilobotics website (https://www.kilobotics.com/docs/index.html) where the whole set of functions are explained. Only the main function are explained below.

- **Kilo_init**()

  Initialize Kilobot hardware.

- **set_motors**(*cw_motor*,*ccw_motor*)

  Set motor speed PWM values for motors between 0 (off) and 255 (full on) for *cw_motor* and *ccw_motor*

  Examples:
  - for moving forward: **set_motors(**kilo_straight_left, kilo_straight_right**);**
  - for turning left: **set_motors(**kilo_turn_left, 0**);**
  - for turning right: **set_motors(**0, c kilo_turn_right**);**

  **Note that there are 4 calibration values used with the motors**

  kilo_turn_left: value for left motor to turn the robot ccw in place (note: right motor should be off)

  kilo_turn_right: value for right motor to turn the robot cw in place (note: left motor should be off)

  kilo_straight_left: value for the left motor to move the robot in the forward direction

  kilo_straight_right: value for the right motor to move the robot in the forward direction

- **delay**(*x*)

  Busy wait for *x* milliseconds, interrupts can still trigger

  Example:

  **delay(***250***);**

- **estimate_distance** (*x*)

  Estimate distance in mm based on signal strength measurements.

  Example:

  **distance = estimate_distance(distance_measurement);**

- **set_color(***r*,*g*,*b***)**

  Set LED color, values can be from 0(off)-3(brightest)

  Example:

  **set_color(***1*,*1*,*1***);**

- **get_voltage()**

  Read the amount of battery voltage.

  Example:

  **current_voltage = get_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();**

- **get_temperature()**

  Read the temperature of the Kilobot.
  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_temperature = get_ temperature ();**

### 4.3.8 Motors calibration

If you see that the robot is no more moving forward and does not rotate well anymore, you can try to recalibrate its motors with the instructions below. But as already said before, the Kilobot should be operated on a smooth, flat surface to ensure proper robot mobility.

Only one Kilobot can be calibrated at the same time.

1. Place the Kilobot in PAUSE mode
2. Open the KiloGUI interface, then open the Calibration mode

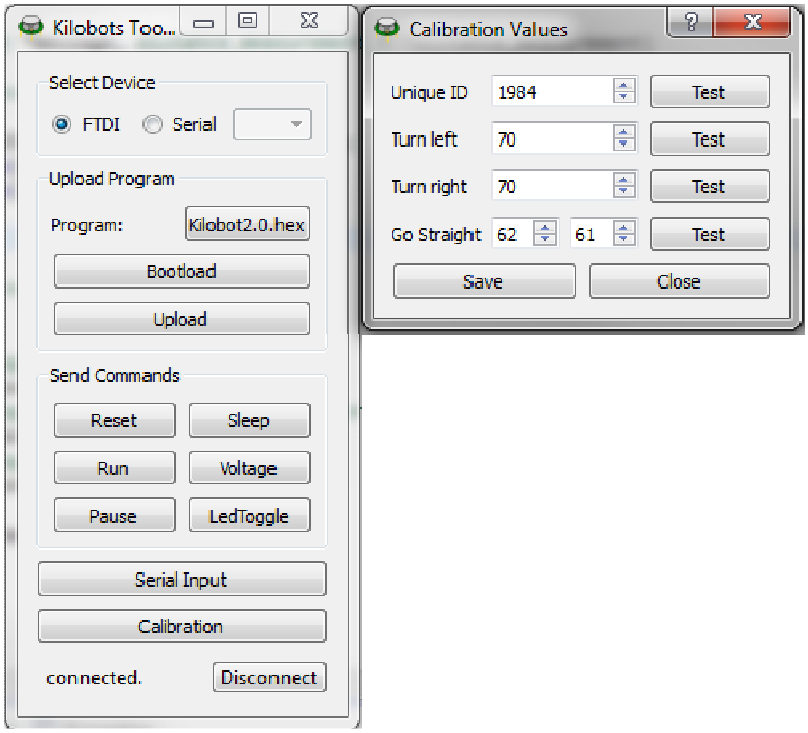



***Figure 4.15: Calibration mode***

3. The first line (***Unique ID***) can be used to save an ID in your Kilobot. This can be useful if you want to save all calibration value for each Kilobot.
4. The second line (***Turn Left***) will configure the ***kilo_turn_left*** parameter to set the CCW movement. Set a value for the motor (approximately 70) and press the Test button. The Kilobot must start turning left. Adjust the value to obtain a smooth move.
5. Do the same as explain in point 4, for the right motor (***Turn Right*** line).
6. Next is to set the straight move parameters. Start with the same value for the left and right motor (approximately 60) and press the Test button. Now adjust the two value to move the Kilobot as straight as possible.
7. Finally, when all the parameters are fine, you can press the ***Save*** button to write the parameters in the EEPROM of the Robot.

### 4.3.9 Simulation

You can simulate the Kilobot robot with V-REP, a power physics simulator software. An educational version is available for free at: http://www.coppeliarobotics.com/downloads.html

The Kilobot model is already available for V-REP inside the simulator.

All the functions explained in chapter 4.3.7 “

Kilobots API" are implemented.

- **get_ambient_light()** function is implemented only in the following because it takes 1.5 more time of simulation:

  scene:

  http://ftp.k-team.com/V-REP/models/Kilobot/scenes/Kilobots_light_intensity.ttt

  model:

  http://ftp.k-team.com/V-REP/models/Kilobot/models/Kilobot_light_intensity.ttm

- For the **_delay_ms**(*x*) function, as it must be non-blocking, you need to implement it like this for example **_delay_ms**(*50*): use a state variable (here **substate**).

```
substate=0
…

function user_prgm()
    …
    -- wait 50 ms:
    if (substate==0) then
        delay_start=simGetSimulationTime()
    elseif (substate==1) then
        if (_delay_ms(50)==1) then
            -- waited
            substate=0
            ----------------------
            -- here put the following of your code
            ----------------------
        end
    …
```

You can find some V-REP scenes with Kilobot examples there:

http://ftp.k-team.com/V-REP/models/Kilobot/scenes

| | | |
|---|---|---|
| Kilobots_messages.ttt | : | 5 robots move and display their distances to each other's with their led. |

| | | |
|---|---|---|
| Kilobots_ messages.ttt | : | 4 robots send messages to each other to determinate the one that has the biggest id number. |
| Kilobots_ around.ttt | : | robots move around other static robots, keeping distance. |

# 5. ANNEXES

## 5.1 Examples of source code

These examples of source code have different purposes described in their titles.

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

```
void setup()
{

        // Initialize message:
        // The type is always NORMAL.
         message.type = NORMAL;
        // Some dummy data as an example.
        message.data[0] = 0;
        // It's important that the CRC is computed after the data has been set;
        // otherwise it would be wrong and the message would be dropped by the
        // receiver.
        message.crc = message_crc(&message);

}


…
// Register the message_tx callback function.
kilo_message_tx = message_tx;
// Register the message_tx_success callback function.
 kilo_message_tx_success = message_tx_success;
//check for message
 if (new_message == 1)
{
        new_message = 0;
        set_color(1,1,1);//turn RGB LED white
        printf(Mess0);//send first byte of received message over serial debug cable
        printf(Mess1);//send second byte of received message over serial debug cable
        printf("          ");
        delay(10);//wait 10 ms
        set_color(0,0,0);//turn RGB LED off

}
```

### 5.1.2 Simple movement

```
void setup()
{
}

void loop()
{
    // Set the LED green.
    set_color(RGB(0, 1, 0));
    // Spinup the motors to overcome friction.
    spinup_motors();
    // Move straight for 2 seconds (2000 ms).
    set_motors(kilo_straight_left, kilo_straight_right);
    delay(2000);

    // Set the LED red.
    set_color(RGB(1, 0, 0));
    // Spinup the motors to overcome friction.
    spinup_motors();
    // Turn left for 2 seconds (2000 ms).
    set_motors(kilo_turn_left, 0);
    delay(2000);

    // Set the LED blue.
    set_color(RGB(0, 0, 1));
    // Spinup the motors to overcome friction.
    spinup_motors();
    // Turn right for 2 seconds (2000 ms).
    set_motors(0, kilo_turn_right);
    delay(2000);

    // Set the LED off.
    set_color(RGB(0, 0, 0));
    // Stop for half a second (500 ms).
    set_motors(0, 0);
    delay(500);
}

int main()
{
    kilo_init();
    kilo_start(setup, loop);

    return 0;
}
```

# 6. WARRANTY

K-TEAM warrants that the Kilobot 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 Kilobot used (version)
- the kernel version of the Kilobot
- the programming environment of the 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 Kilobot 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 labour 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.
Z.I. PLANS-PRAZ
1337 VALLORBE
SWITZERLAND