

MOWAYDUINO

QUICK GUIDE



Copyright (c) 2013 Bizintek Innova, S.L.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 2.0 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Index

| | | |
|------|--------------------------------------|----|
| 1. | Installing the mOwayduino Pack | 4 |
| 2. | Installing the drivers | 4 |
| 3. | Programming mOwayduino | 4 |
| 3.1. | Programming with Arduino IDE | 5 |
| 3.2. | Programming with Scratch | 6 |
| 4. | mOwayduino libraries | 8 |
| 4.1. | Sensors library | 8 |
| 4.2. | Motors library | 9 |
| 4.3. | RF library | 10 |
| 4.4. | Camera library | 10 |
| 5. | Scratch commands | 11 |
| 5.1. | Variables | 11 |
| 5.2. | Commands | 12 |
| 5.3. | Sensors | 12 |
| 6. | Wifi module | 13 |
| 7. | Mowayduino Cam | 20 |

1. Installing the mOwayduino Pack

The mOwayduino Pack includes all the libraries and software in order to program and control the mOwayduino robot. Download the mOwayduino Pack Installer from:

www.mowayduino.com

Once it is downloaded, please follow the steps described on the installer. When installing the mOwayduino Pack, it is important to select the folder where Arduino IDE is installed on the PC (where “arduino.exe” is located). You can get the Arduino IDE from this link:

<http://arduino.cc/en/Main/Software>

NOTE: When mOwayduino Pack is installed in Arduino IDE directory, the original “boards.txt” file of Arduino IDE is renamed as “boards_BACKUP.txt”. So that, if the user modified this file, the changes are saved in that backup file.

2. Installing the drivers

Once mOwayduino Pack is installed, connect the robot to PC. You can find the drivers in the “drivers” folder of Arduino IDE:

“your_path \ arduino-1.0.5 \ drivers”

The drivers are also installed in mOwayduino folder in “Program Files” folder. Here you can find also the RFUSB and the Camera Board drivers:

“C:\ Program Files \ mOwayduino \ mOwayduino Drivers”

3. Programming mOwayduino

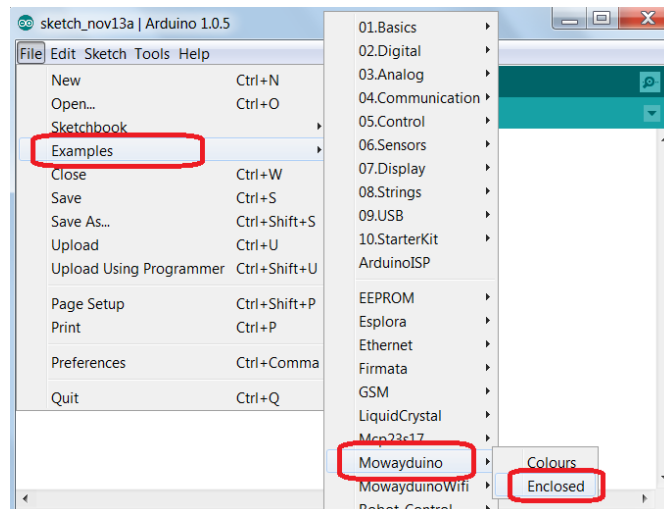
You can program mOwayduino with:

- Arduino IDE
- Scratch

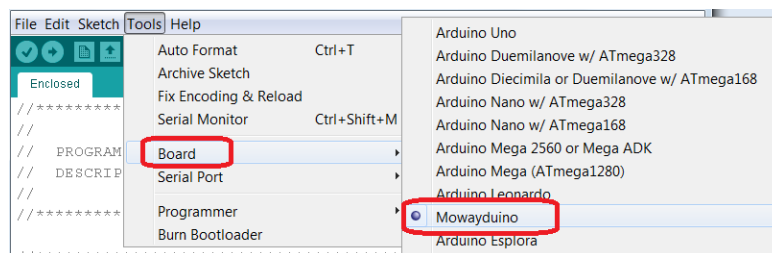
3.1. Programming with Arduino IDE

In order to program with Arduino IDE, follow these steps:

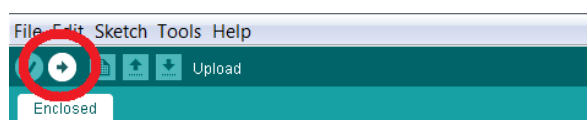
- 1) Connect mOwayduino to PC with USB cable
- 2) Open Arduino IDE
- 3) Open a mOwayduino program, for example, “Enclosed”



- 4) Select “mOwayduino” board



- 5) Select the serial port of mOwayduino (value may vary)
- 6) Press “Upload” button



This program will make the robot to be inside of a black circle painted over a white surface.

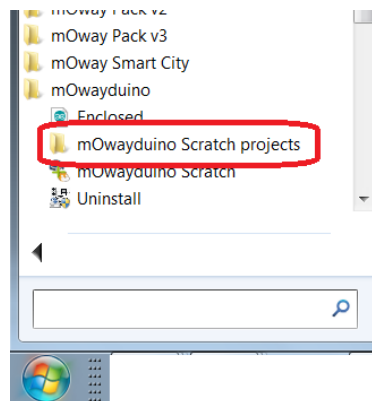
3.2. Programming with Scratch

You can begin using Scratch for mOwayduino with this example:

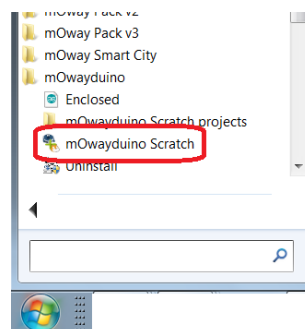
- 1) Connect RFUSB device to PC.



- 2) Open the example program for mOwayduino, in “All the programs -> mOwayduino -> mOwayduino Scratch Projects -> mowayduino_rc.sb”.



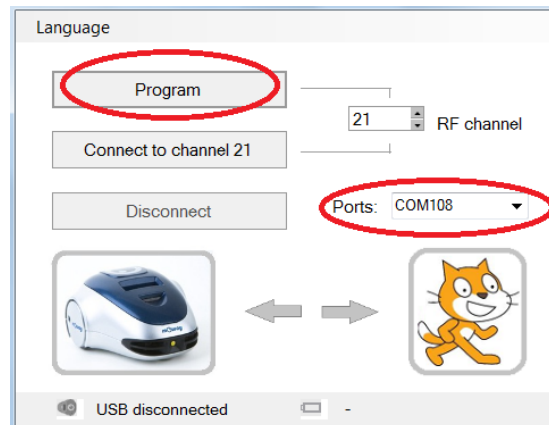
- 3) Open “mOwaduinoScratch”, which is an interface between Scratch environment and mOwayduino robot (“All the programs -> mOwayduino -> mOwayduinoScratch”).



- 4) Connect the robot to PC and program it. The correct COM Port has to be selected (it usually appears only one). In case the programming fails, reconnect

the robot to PC, select the correct COM port and program it (COM port number may vary).

NOTE: You can change the RF channel before programming in case there are more than one robot, in order to avoid interferences between them.



- 5) Once programmed, press “Connect to channel...”.
- 6) Now you can use the Scratch programming.



In this example, you can control the mOwayduino movements with the arrow keys.

4. mOwayduino libraries

You can find the mOwayduino libraries by default in the Arduino installation folder, usually in:

“C:\ Program Files \ Arduino \ libraries \ Mowayduino”

Here you can find all the files for programming mOwayduino. These files contain the explanation and usage of each function.

| mOwayduino device | Header file | Code file |
|----------------------------|-------------------|---------------------|
| Camera control library | Camera.h | Camera.cpp |
| Movements | Motor.h | Motor.cpp |
| mOwayduino pins | Mowayduinolo.h | - |
| RF communication | MowayduinoRf.h | MowayduinoRf.cpp |
| Sensors, LEDs, Speaker,... | MowayduinoRobot.h | MowayduinoRobot.cpp |

4.1. Sensors library

| Function | Description | Value |
|-------------------------|---|--|
| void beginMowayduino(); | Configure robot sensors | - |
| void readObstacle(); | Read the 4 obstacle sensors. The result is saved in “Obstacles” array | 0: No obstacle 1024: Closest obstacle |
| void readLine(); | Read the 2 line sensors. The result is saved in “Line” array. | 0: White 1: Black |
| int readLight(); | Read light sensor | 0: No light 1024: High brightness |
| int readBatt() | Read battery level | 0 – 100% |
| int readMic(); | Read microphone value | 0: Silence 1024: Loud noise |
| void Speakeron(); | Activates the speaker | - |
| void Speakeroff(); | Deactivates the speaker | - |
| void readAcc(); | Read accelerometer. The result is saved | -2g – 2g for each axis |

| | | |
|------------------|---|-----------|
| | in “Accelerometer” array. | (X, Y, Z) |
| void Redon(); | Turns on the RGB red LED | - |
| void Redoff() | Turns off the RGB red LED | - |
| void Greenon() | Turns on the RGB green LED | - |
| void Greenoff() | Turns off the RGB green LED | - |
| void Blueon() | Turns on the RGB blue LED | - |
| void Blueoff() | Turns off the RGB blue LED | - |
| void Fronton(); | Turns on the front LED | - |
| void Frontoff(); | Turns off the front LED | - |
| void Brakeon(); | Turns on the brake LED | - |
| void Brakeoff(); | Turns off the brake LED | - |
| void Ledsoff(); | Turns off all the LEDs | - |
| void TuneAcc(); | Calibrate accelerometer values. Place mOwayduino over a flat surface without movement. When the values are calibrated, the green LED will blink and the speaker will make a beep. | - |

4.2. Motors library

| Function | Description |
|---|--|
| void beginMotor(); | Configure motors |
| void Straight(byte speed, int distance); | Go straight (distance is optional) |
| void Back(byte speed, int distance); | Go back (distance is optional) |
| void TurnRight(byte speed, int angle); | Turn right over the wheel (angle is optional) |
| void TurnLeft(byte speed, int angle); | Turn left over the wheel (angle is optional) |
| void RotateRight(byte speed, int angle); | Turn right over the centre (angle is optional) |
| void RotateLeft(byte speed, int angle); | Turn left over the centre (angle is optional) |
| void Move(int left_speed, int right_speed); | Move the wheels independently |

| | |
|-------------------|---|
| void Stop(); | Stop |
| void MotorTune(); | This function calibrates the motors (NOTE 1). |

NOTE 1: Call this function if mOwayduino doesn't move straight when it should (for example, in a "Straight(speed)" command, for moving straight forward). When this function is called, the robot will move forward 50cm while it calibrate the encoder reading of the wheels. It will repeat until the motors are calibrated. Then it stores the values on its EEPROM memory, so it is not necessary to call this function again.

You can use the "TuneMotors" program included in "*Examples -> Mowayduino -> TuneMotors*" in Arduino IDE.

4.3. RF library

| Function | Description | Parameters |
|---|--------------|--|
| unsigned char beginRF(unsigned char chan, unsigned char add); | Configure RF | - RF channel - Robot address |
| unsigned char On() | Turn RF on | - |
| unsigned char Off() | Turn RF off | - |
| unsigned char Send(unsigned char RF_DIR_OUT, unsigned char RF_DATA_OUT[]) | Send data | - Address to send data to - Array of 8 bytes to be sent |
| unsigned char Receive(unsigned char* RF_DIR_IN, unsigned char* RF_DATA_IN); | Receive data | - Address that sent the data - 8 bytes of received data |

4.4. Camera library

| Function | Description | Parameters |
|-----------------------------|-------------------|-----------------------------|
| void beginCamera() | Configures camera | - |
| void cameraOn(byte channel) | Turns on camera | Transmission channel (1 -4) |
| void cameraOff() | Turns off camera | - |

5. Scratch commands

mOwayduino robot is controlled by means of “variables” and “commands”. Changing the value of the variables allows to control the elements of the robot, for example the LEDs, motors, speaker, etc. It is also possible to send commands to the robot to make easier some task, for example following a black line, pushing objects, etc.

You can also use the values of mOwayduino sensors for developing your programs.

5.1. Variables

| Element | Variable name | Values |
|---------|---------------|---|
| LEDs | front_led | 0: off 1: on |
| | brake_led | 0: off 1: on |
| | top_led | r: red g: green b: blue y: yellow t: turquoise p: purple w: white |
| Motors | left_motor | 80 – 200: Speed of the left motor |
| | right_motor | 80 – 200: Speed of the left motor |
| Speaker | speaker | 0: off 1: on |
| Other | angle | 0 – 360: rotation angle for “turn_left” and “turn_right” commands |

5.2. Commands

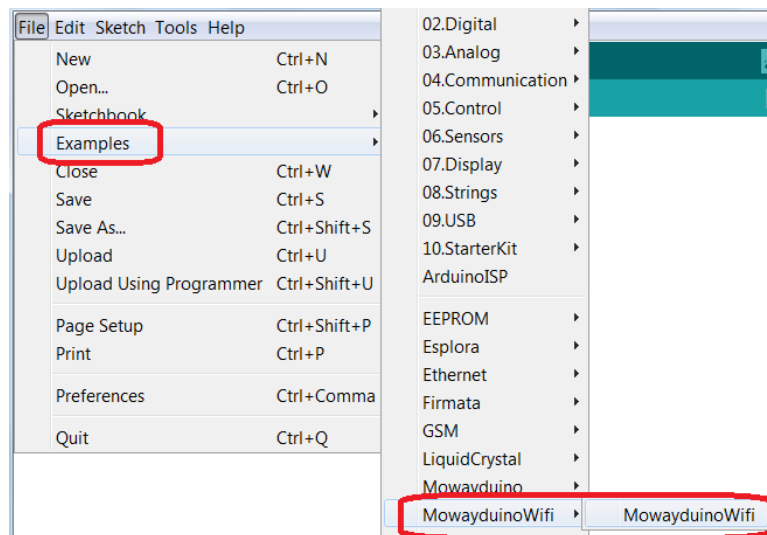
| Command | Description | Used variables |
|-------------------|---|----------------|
| enclosed | Subprogram for enclosing mOway into a black circle. It goes forward when it is on a white surface and it turns around when it reaches a black line. | - |
| push | Subprogram for pushing objects. mOway searches the objects and pushes them. | - |
| defender | Subprogram for pushing objects out of a black circle. It puts together the “enclosed” and “push” subprograms. | - |
| follow_line_left | Subprogram for following a line on the left side. | - |
| follow_line_right | Subprogram for following a line on the right side. | - |
| turn_left | Subprogram for following a line on the right side. | angle |
| turn_right | Subprogram for following a line on the right side. | angle |
| turn_around | Subprogram for turning 180 degrees | |

5.3. Sensors

| Variable | Description | Range of values |
|-----------------------|---------------------------------|----------------------|
| Obstacle Side Left | Obstacle Side Left Sensor | 0 – 255 |
| Obstacle Center Left | Obstacle Center Left | 0 – 255 |
| Obstacle Center Right | Obstacle Center Right Sensor | 0 – 255 |
| Obstacle Side Right | Obstacle Side Right Sensor | 0 – 255 |
| Line Left | Line left Sensor | 0: white 1: black |
| Line Right | Line right Sensor | 0: white 1: black |
| Light | Ambient Light Sensor | 0 – 255 |
| Microphone | Noise level | 0 – 255 |
| X-Axis g | Acceleration in g in the axis X | -2.0 to 2.0 g |
| Y-Axis g | Acceleration in g in the axis Y | -2.0 to 2.0 g |
| Z-Axis g | Acceleration in g in the axis Z | -2.0 to 2.0 g |

6. Wifi module

The Wifi module allows to connect mOwayduino to Wi-Fi nets. You can find an example program in “Examples -> MowayduinoWifi”:



Once this example is opened, you can modify it to make the robot to be a web server, to publish tweets or to connect to a socket. The Wifi module has to be connected to mOwayduino for using these examples.



6.1. Web server example

In order to run a web server for controlling mOwayduino, follow these steps:

- In “MowayduinoWifi.ino” file (line 6) uncomment “#define WEB_SERVER”
- Change the “apps-conf.h” file as it is described below

```

// #define TWITTER EXAMPLE
#define WEB_SERVER
/* IMPORTANT */ /* COMMENT #define APP_SOCKAPP and UNCOMMENT #define APP_WISERVER
in apps-conf.h in arduino/libraries/MowayduinoWifi folder */
  
```

- 1) In file “MowayduinoWifi.h” (lin 17) write the Wi-Fi net to connect to (“ssid[]”) and its password (“security_passphrase[]”). Also change IP configuration if needed.

```

unsigned char local_ip[] = {192,168,1,25}; // IP address of mOwayduino
unsigned char gateway_ip[] = {192,168,1,1}; // router or gateway IP address
unsigned char subnet_mask[] = {255,255,255,0}; // subnet mask for the local network
const prog_char ssid[] PROGMEM = {"*****"}; // Replace "****" for your net name (max 32 byt
unsigned char security_type = 2; // 0 - open; 1 - WEP; 2 - WPA; 3 - WPA2
// WPA/WPA2 passphrase
const prog_char security_passphrase[] PROGMEM = {"*****"}; // Replace "****" for your net password
  
```

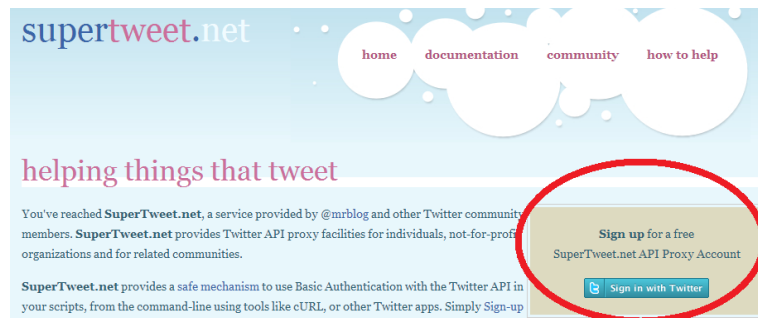
Once the robot is programmed, it will connect to the chosen Wi-Fi net (it can take some time). Then typing the chosen IP in a web browser will display a basic control of the robot.



6.2. Twitter example

In order to send tweets from mOwayduino:

- 1) Create an account on www.supertweet.net. This web acts like an interface between mOwayduino robot and *Twitter* (a *Twitter* account is needed).



- 2) Once logged in, create password on *supertweet* (different from your twitter password)

Access to the SuperTweet.net Twitter API proxy requires the use of special credentials that are associated with your account. Welcome, SMiniro

g+1 +9 Recomendar esto en

Access Credentials

| Username | Secret | Status |
|-------------|---------|---------------------|
| SMinirobots | Not Set | Inactive (Activate) |

- 3) In “MowayduinoWifi.ino” file (line 6) uncomment “#define WEB_SERVER”
- 4) Change the “apps-conf.h” file as it is described below

```

// #define TWITTER EXAMPLE
#define WEB_SERVER
/* IMPORTANT */ /* COMMENT #define APP_SOCKAPP and UNCOMMENT #define APP_WISERVER
in apps-conf.h in arduino/libraries/MowayduinoWifi folder */

```

- 5) In file “MowayduinoWifi.h” (lin 17) write the Wi-Fi net to connect to (“ssid[]”) and its password (“security_passphrase[]”). Also change IP configuration if needed.

```

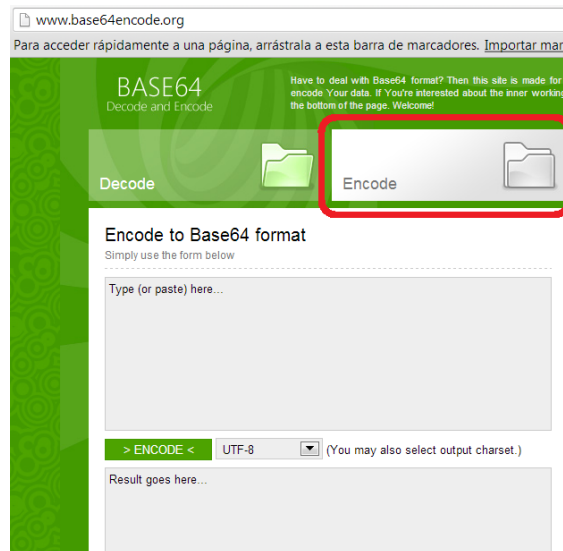
unsigned char local_ip[] = {192,168,1,25}; // IP address of mOwayduino
unsigned char gateway_ip[] = {192,168,1,1}; // router or gateway IP address
unsigned char subnet_mask[] = {255,255,255,0}; // subnet mask for the local network
const prog_char ssid[] PROGMEM = {"*****"}; // Replace "****" for your net name (max 32 byt
unsigned char security_type = 2; // 0 - open; 1 - WEP; 2 - WPA; 3 - WPA2

// WPA/WPA2 passphrase
const prog_char security_passphrase[] PROGMEM = {"*****"}; // Replace "****" for your net password

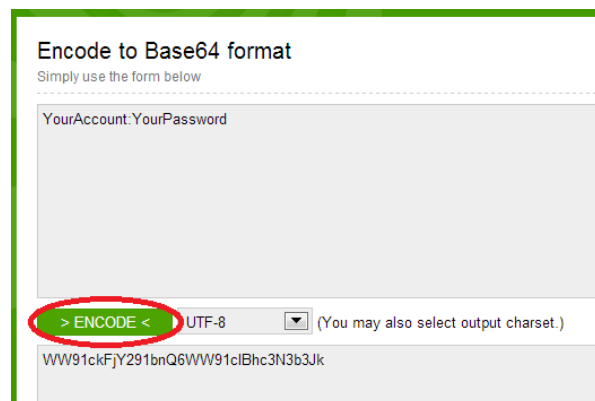
```

- 6) Also change the *supertweet* authorization in “MowayduinoWifi.h” (lin 58). This authorization is your user name and your password that you created on

supertweet. To do so, go to "<http://www.base64encode.org/>" and encode the authorization to Base64 format. Select the "Encode" option.



- 7) Write the authorization. It has to have a format like "YourUsername:YourPassword". The press on "> ENCODE <".



- 8) Copy the result and paste it on "MowayduinoWifi.h", on line 58.

```
const char supertweetauth[] = "*****"; // Replace '*****' for "user:password" of supertweet in Base64
// IP Address for www.supertweet.net
uint8 supertweetip[] = {50,18,251,175};
// A request that sends mOwayduino light sensor value to "supertweet"
POSTrequest twittermowayduino(supertweetip, 80, "api.supertweet.net", "/1.1/statuses/update.json",tweet_write);
```

A new tweet will be published on your twitter displaying the value of mOwayduino sensors.



6.3. Socket example

In order to communicate with a socket, follow these steps:

- 1) In “MowayduinoWifi.ino” (line 10), comment “TWITTER_EXAMPLE” and “WEB_SERVER” definitions, and uncomment “SOCKET” definition. Also modify “apps-conf.h” file as it is marked below.

```
//#define TWITTER_EXAMPLE
//#define WEB_SERVER
/* IMPORTANT */ /* COMMENT #define APP_SOCKAPP and UNCOMMENT #define APP_WISERVER
in apps-conf.h in arduino/libraries/MowayduinoWifi folder */

#define SOCKET
/* IMPORTANT */ /* UNCOMMENT #define APP_SOCKAPP and COMMENT #define APP_WISERVER
in apps-conf.h in arduino/libraries/MowayduinoWifi folder
UNCOMMENT #define socket in socketapp.c */
```

- 2) In “socket.c” file (line 45) uncomment the “SOCKET” definition.

```
* We define the application state (struct socket_app_state) in the
* socketapp.h file, so we need to include it here. We also include
* uip.h (since this cannot be included in socketapp.h) and
* <string.h>, since we use the memcpy() function in the code.
*/
#define SOCKET
#ifdef SOCKET

#include "socketapp.h"
#include "uip.h"
#include <string.h>
```

- Open a socket communication. You can do this for example with *putty.exe*, a software for socket communication. You can download it from:

<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

Binaries

The latest release version (beta 0.63). This will generally be a version I think is reasonably development snapshot (below) to see if I've already fixed the bug, before reporting it to me.

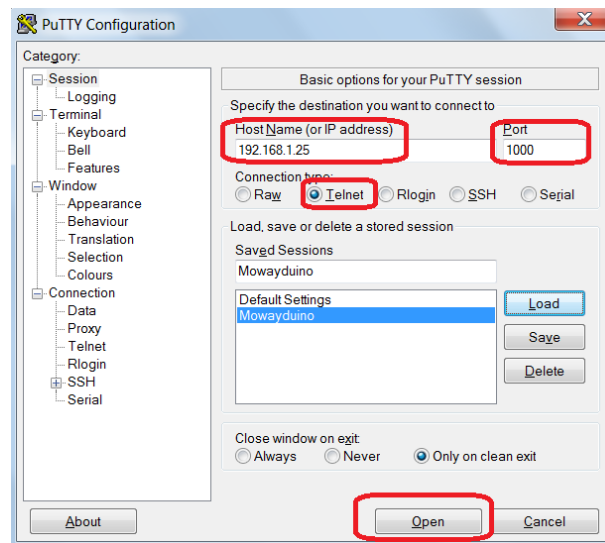
For Windows on Intel x86

| | | | |
|-----------|------------------------------|-----------------------------|---------------------------|
| PuTTY: | putty.exe | (or by FTP) | (RSA sig) |
| PuTTYtel: | puttytel.exe | (or by FTP) | (RSA sig) |
| PSCP: | pscp.exe | (or by FTP) | (RSA sig) |
| PSFTP: | psftp.exe | (or by FTP) | (RSA sig) |
| Plink: | plink.exe | (or by FTP) | (RSA sig) |
| Pageant: | pageant.exe | (or by FTP) | (RSA sig) |
| PuTTYgen: | puttygen.exe | (or by FTP) | (RSA sig) |

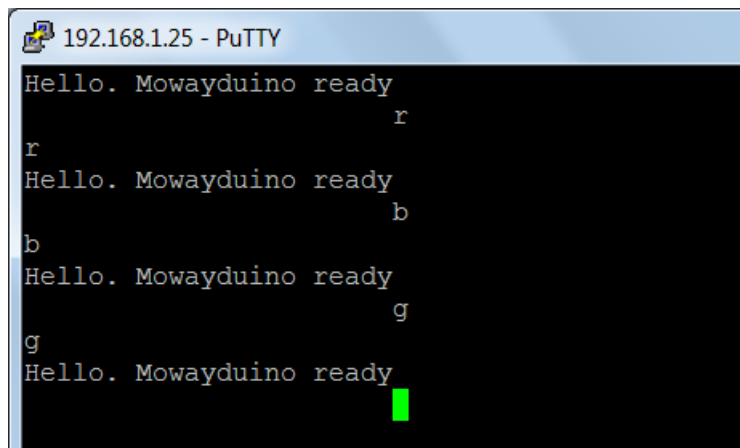
A .ZIP file containing all the binaries (except PuTTYtel), and also the help files

| | | | |
|-----------|---------------------------|-----------------------------|---------------------------|
| Zip file: | putty.zip | (or by FTP) | (RSA sig) |
|-----------|---------------------------|-----------------------------|---------------------------|

- Once downloaded, open a Telnet socket in port 1000 with the chosen IP.



- You will be able to control mOwayduino LEDs by pressing “g” (green), “r” (red) and “b” (blue).

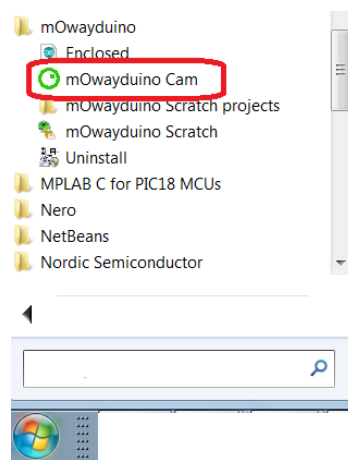


```
192.168.1.25 - PuTTY
Hello. Mowayduino ready
r
Hello. Mowayduino ready
b
Hello. Mowayduino ready
g
Hello. Mowayduino ready
```

7. Mowayduino Cam

“Mowayduino Cam” software allows to watch the images from the camera of mOwayduino. In order to watch these images, mOwayduino has to be programmed for activating the camera. It is also necessary to connect the camera to the expansion connector of the robot and to connect the Camera Board receptor to an USB port of PC.

You can launch “Mowayduino Cam” from “All programs -> mOwayduino -> Mowayduino Cam”:



Once the Camera Board is connected to PC, press “Start” button and switch on mOwayduino to activate the camera:

