Global Specialties®

RP6v2 CONTROL M256 WiFi Expansion Module

globalspecialties.com

1-800-572-1028
22820 Savi Ranch Parkway   Yorba Linda, CA 92887

RP6v2 CONTROL M256 WiFi

Instruction Manual

- English -

Version RP6M256-EN-20130312

IMPORTANT INFORMATION!
Please read carefully!

Before you start operating this RP6 expansion module, you must read this manual AND the RP6 ROBOT SYSTEM manual completely! The documentation contains information about how to operate the devices properly and how to avoid dangerous situations! Furthermore the manuals provide important details, which may be unknown to average users. The RP6 CONTROL M256 manual is only supplementary documentation!

Paying no attention to this manual will cause a loss of warranty! Additionally, Global Specialties cannot be made responsible for any damages caused by neglecting the manual's instructions!

Please pay special attention to the chapter "Safety instructions" in the RP6 ROBOT SYSTEM manual!

Limited One-Year Warranty

Global Specialties warrants to the original purchaser that its products and the component parts thereof, will be free from defects in workmanship and materials for a period of one year from date of purchase.

Global Specialties will, without charge, repair or replace, at its option, defective product or component parts. Returned products must be accompanied by proof of the purchase date in the form of a sales receipt.

Exclusions: This warranty does not apply in the event of misuse or abuse of the product or as a result of unauthorized alterations or repairs.
Global Specialties shall not be liable for any consequential damages, including without limitation, damages resulting from loss of use. Some states do not allow limitations of incidental or consequential damages. So the above limitations or exclusions may not apply to you.

This warranty gives you specific rights and you may have other rights, which vary from state-to-state.

Symbols

The following Symbols are used in this manual:

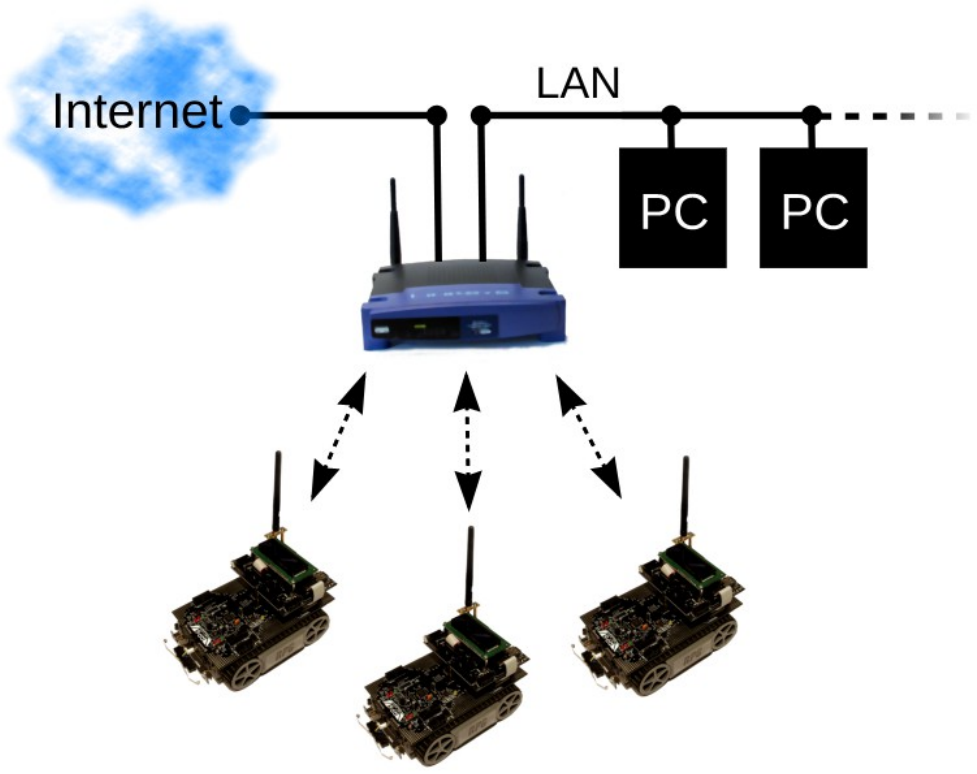
The “Attention!” Symbol is used to mark important details. Neglecting these instructions may damage or destroy the robot and/or additional components and you may risk your own or others health!

The "Information" Symbol is used to mark useful tips and tricks or background information. In this case the information is to be considered as “useful, but not necessary”.

Contents

1. The RP6v2 M256 WIFI Expansion Module

The RP6v2 Control M256 WIFI expansion module (Part # RP6v2-M256-WIFI or in short RP6-M256) allows you to integrate one or more RP6 and RP6v2 robots in a wireless computer network, to control them remotely and to transfer telemetry data. If necessary the robots can also retrieve data from the internet on their own.



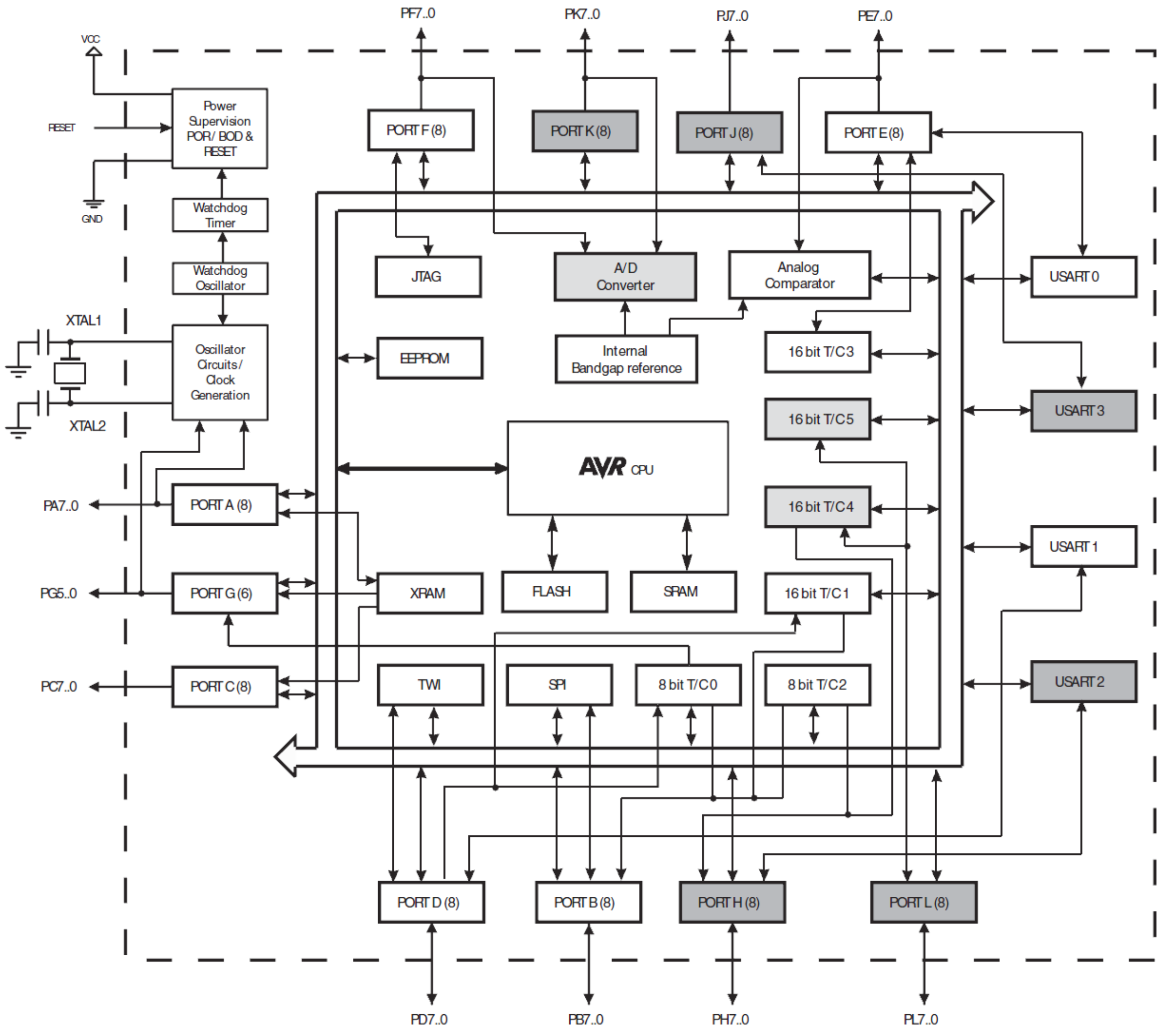
The module contains the biggest microcontroller from the ATMEGA series of Atmel. With 256KB flash ROM (of which 8KB are dedicated to the bootloader) and 8KB SRAM, the ATMEGA2560 contains 8x more flash memory and 4x more SRAM than the controller on the RP6 mainboard. Thus it offers more than enough space for very complex user programs. The user-friendly software library is compatible with the other AVR processors of the RP6 such that is not necessary to start learning everything again.

A specialty of the module is the highly energy efficient 802.11b/g WLAN module with built-in processor that is soldered directly on the PCB. The internal processor is managing all WLAN functions, the TCP/IP stack and the AES/WPA2 encryption. This relieves the AVR main processor and simplifies the control for the user (simple text commands, control via a serial interface). Immediately after power on, the module can connect automatically to the WLAN network within a second (typical) and start data transfer.

The energy consumption of the whole module is very low. With active 500kbit/s WLAN data transfer, maximum transmission performance and a running program in the ATMEGA2560 processor, the RP6v2-M256-WiFi module requires less than 460mW in total. Therefore it is ideally suited for battery powered applications like mobile robots.

The clock frequency of the ATMEGA2560 is twice as fast (16MHz) as the controller of the RP6 main board. At the same time there is more free calculation time on the RP6-M256 because the controller is not busy with motor regulation, ACS, IRCOMM, etc.

This page shows a simplified block diagram with the peripherals of the ATMEGA2560 controller. Among others, we recognize the AVR CPU core in the center with flash ROM and SRAM memory together with bus connection to the peripherals. Around it there are the numerous 8 bit I/O ports, 6 timers, 4 USARTs (serial interfaces or two dual SPI / UART ports,), SPI, I^{2}C Bus (TWI), A/D converter, comparator and EEPROM that are all linked together via a bus. The clock generator and reset unit are also very important.



In addition, there are direct links from the peripherals such as timers or serial interfaces to the I/O ports such that these can be directly connected to external units. I.e. for the timers the PWM outputs or for the serial interfaces the transmission and reception signals can be directly linked to the I/O ports.

The RP6-M256 module offers more than 60 free 5V I/O pins (+6 on the XBUS connectors and +2 on the UART connector, i.e. up to a total of 68) on easy to use 2.54mm grid connectors. These feature a multitude of alternate functions like 16x A/D converter channels, 12x hardware PWM outputs (e.g. for servo motors, dimmable LED searchlights, piezo buzzers and others), one 8-bit display port, 2x USART/SPI ports, 22x external interrupts, 4x timer capture inputs, one analog comparator, I2C bus, USART and one ISP programming connector (shared with a micro SD card slot).
Plus 4x A/D converter channels from the WLAN module. On top of that are one micro SD card slot for flash cards as mass storage device, two push-buttons and seven status LEDs (3x WLAN module, 4x micro controllers).

Similar to the other RP6 expansion modules (RP6-M32 and RP6-M128), you can connect an LCD display to show the current status and sensor values. The numerous I/O ports allow to add lots of user interface elements like additional buttons.

A 10cm tall 2.4GHz antenna with 2dBi gain, 15cm u.FL to RP-SMA pigtail cable and mounting hardware complete the included equipment. The mounting hardware allows experienced users to install the antenna at other locations.

Due to the internal bootloader, new software can be easily uploaded wirelessly via the WLAN link into the module. Once it has been properly set up, a new program can be loaded into the ATMEGA2560 with one mouse click. The transmission speed is quite similar to the wired USB link (with good environmental conditions / little interference).

The WLAN module can be configured via the RobotLoader software on the host PC by using the RP6 USB interface. This can be done either through a user interface or per text commands sent via the terminal. Once properly configured, the microcontroller program can simply send text messages via the serial interface just as if it was linked to the PC via a cable (of course some small delays or interruptions may occur compared to a wired transmission).

1.1. Safety Instructions

Before you start with the RP6-M256, you should familiarize yourself with the robot itself and try all its example programs WITHOUT the installed RP6-M256 expansion module! This manual is a small addition to the big RP6 manual. Please read it thoroughly before you do anything with the RP6-M256! **The safety instructions in the RP6 manual apply in addition to these, especially the instructions about ESD and the risk of injuries due to sharp pins on the PCB.**

Important hint for beginners:

Programs written for the RP6-M256 DO NOT run properly on the microcontroller of the base unit and the other way round (completely different pin assignment, clock frequency and totally different hardware environment, even different processor in this case)! The same applies for all other expansion modules.

CAUTION:

If you upload a program into the wrong controller, this or the controlled circuits might well be damaged! E.g. if an I/O pin is normally used as an input but the program for the wrong controller configures it as an output, it could get overloaded because of the connected circuit!

Normally nothing serious will happen if you make a mistake but this can not be guaranteed. The RobotLoader can not distinguish the programs because the hexfiles are all built up in the same way. Therefore it will not prevent you from uploading the programs into the wrong controller. **Make use of the function of RobotLoader to define different categories! For every expansion module you can create its own category!** For the RP6-M256 there is an extra hexfile list in the WIFI Loader tab of the RobotLoader software.

CAUTION:

Unplug the antenna cable from the small connector on the PCB (u.FL) only if it is absolutely necessary! Because of the small dimensions, this connector does not support many mating cycles. Handle the plug with care!

The big RP-SMA connector on the other side of the cable is much more reliable! Here you can screw and unscrew the antenna many times without any problem although it is advisable to limit the mating cycles as much as possible as wearout will reduce the signal quality.

1.2. Technical Support

You may contact our support team as follows (**please read this manual completely before contacting the support!** Reading the manual carefully will answer most of your possible questions already! Please also read the appendix for troubleshooting):

Global Specialties

22820 Savi Ranch Parkway
Yorba Linda, CA 92887 USA

Tel No: (800) 572-1028
Fax No: (215) 830-7370

E-Mail: `info@globalspecialties.com`

www.globalspecialties.com

All **software updates, new versions of this manual** and further informations will be published on this site.

1.3. Scope of Delivery

You should find the following items in the carton of the RP6v2 CONTROL M256 WIFI:

- Completely assembled RP6-M256 module
- 1 x 10cm 2.4GHz WLAN antenna
- 1 x 15cm pigtail cable, u.FL → RP-SMA
- Mounting material for the antenna
  The antenna socket should be premounted.
  The antenna cable is already connected.
  You just need to screw the antenna onto the RP-SMA connector.
- 4 x 25mm M3 spacer bolts
- 4 x M3 screws
- 4 x M3 nuts
- 2 x adapters with 2x M3 drill for display installation
- 1 x 14-pin ribbon cable
- 1 x RP6v2 CD-ROM with software and instruction manual

Go to our homepage to download updated software versions and manuals. It is recommended to download software and documents from the homepage as the CD-ROM cannot be updated very often!

1.4. Features and Specifications

This chapter gives you an idea of what the RP6 CONTROL M256 offers and is at the same time an introduction to some terms and description of components of the module.

Features, Components and Specifications of the RP6 CONTROL M256 WIFI:

- **High performance Atmel ATMEGA2560 8-bit microcontroller**
  - ◊ Speed 16 MIPS (=16 million instructions per second) at a clock frequency of 16MHz, which is twice as fast as the controller on the RP6 main board.
  - ◊ Memory: 256KB flash ROM, 8KB SRAM, 4KB EEPROM
  - ◊ 6 hardware timers (2x 8 bit, 4x 16 bit)
  - ◊ Freely programmable in C (with WinAVR / avr-gcc)!
  - ◊ Software library compatible with the other RP6 modules
  - ◊ ... and much more (s. Datasheet)!
- **Energy efficient 802.11b/g 2.4GHz WLAN RF module**
  - ◊ Type: Roving Networks RN-171 (full specifications, see data sheet!)
  - ◊ Has its own processor with complete TCP/IP stack and WPA2-AES encryption, simple control via a serial interface
  - ◊ Automatically connects to the WLAN access point within a second (typical) to start immediate data transfer
  - ◊ 10cm long 2.4GHz antenna with 15cm pigtail cable
  - ◊ Distance max. 100m line of sight (free of obstacles, depends on interference level)
  - ◊ Data rate via serial interface 500kbit/s (identical to the bit rate which is used for the bootloader via the USB interface in "high speed" mode). This is more than enough for all kinds of telemetry transmission.
  - ◊ Data rate via WLAN: max. 54Mbit/s (in this application this rate is of course not utilized), standard setting to increase the transmission distance is 6Mbit/s.
  - ◊ Transparent data transmission per TCP (or UDP).
  - ◊ The firmware supports many other protocols (NTP, FTP Client, HTTP Client etc.), you will find more about it in the documentation of the WLAN module.
  - ◊ Adhoc mode can be activated per jumper (and restore the factory settings)
  - ◊ Firmware of the WLAN module can be updated via Internet (FTP)
  - ◊ Level shifters for 3.3V ↔ 5V are contained on the PCB. Therefore the microcontroller can run normally with 5V and is compatible with the RP6 and many normal robot sensors/actuators. 3.3V sensors can of course be connected via more level shifters.
- **Power consumption of the whole RP6-M256 module at 5V is only about 460mW with an active 500kbit/s data transfer** (this value might vary depending on the temperature, use of peripherals and software)

- **More than 60 free 5V I/O ports** on easy to use 2.54mm grid connectors, **with following alternative functions:**
  - ◇ 16x 10 bit A/D converter channels
  - ◇ 12x 16 bit hardware PWM channels
  - ◇ 3x 8 bit hardware PWM channels
  - ◇ 4x timer capture inputs
  - ◇ 8 bit display port, also usable as normal I/O ports
  - ◇ 2x USART/SPI ports, can be used either as Master SPI or as UART
  - ◇ 22x external interrupts on various I/O ports
  - ◇ I2C bus
  - ◇ USART as bootload/programming connector, can also be used for other purposes if the WLAN module is used for the software upload.
  - ◇ ISP programming connector (for advanced users)
  - ◇ JTAG interface (usable via adapter cable, advanced users only)
- **4x A/D converter channels and 3.3V power rail from the WLAN module** are accessible on a 6-pin header. On board voltage divider to increase the voltage range. Normally these channels only have a measurement range from 0 to 0.4V, with 0.5V threshold for wakeup triggers and 1.2V maximum voltage. External voltage dividers extend the measurement range up to 3.12V, with 3.9V wakeup trigger, and 9V maximum voltage (recommended are 5V max like for the other A/D converters). These can be read independently from the microcontroller and also be used to wake up the WLAN module. Voltage divider factor: 0.12821
- **MicroSD card slot for memory cards**
  - ◇ Ideal for data logging, mission data or eventually storage of static data that will be retrieved via WLAN (note: the data rate is of course limited, this is not a full file server)
  - ◇ Compatible with SD and SDHC cards that support SPI mode (should apply to almost all cards)
  - ◇ SDHC is more complex, needs more memory and calculation time, therefore cards up to 2Gbyte are highly recommended. Bigger cards do work fine as well though. This can be configured in the program code.
- **I²C-Bus Expansion connectors**
  - ◇ Can control any number of I²C bus slaves.
  - ◇ The module's MEGA2560 controller may be used as master or slave device. Usually we suggest to use the expansion board controller as master for complete control of the robot and to use the mainboard's controller as slave for motor speed control, ACS, IRCOMM, battery monitoring, etc. in order to disburden the expansion board controller.
- **4x status LEDs connected to the microcontroller**
- **3x status LEDs connected to the WLAN module (communication, activity, error status)**
- **2x buttons**

- **LC-Display port**
  - ◇ Suitable for connecting a 16x2 character LCD. The port also accepts other LCDs such as 16x4, however these must be fastened via two spacer bolts and would protrude on one side... It is recommended to check the dimensions prior to buying and ordering appropriate mounting hardware at the same time! If you use a different display format than 16x2, you might need to make some adjustments in the library to make sure that it works properly (mainly the initialization of the display and eventually the cursor positioning). All example programs are tailor-made for 16x2 character displays.
  - ◇ The display will show e.g. text messages/menus, program status or sensor values.
  - ◇ 8 bit display port with all control signals. Can also be used as normal I/O ports, dual pin configuration with a part of the XMEM interface (AD0-7, ALE, RD, WR, plus A8 and A9 available on an extra 3-pin header).
- **All 4 external interrupts on the XBUS connector are available** (this is not the case on the RP6-M32 Module)**.**
- **USB PC interface connector**
  - ◇ For program upload. This is done exactly as with the robot itself simply and easily via the separate USB interface and the user-friendly RobotLoader software.
  - ◇ Alternatively - or primarily in the case of this module - the on board WLAN module can also be used to upload new programs. This frees up the serial interface for other applications (however during booting the bootloader generates outputs with 38400 bauds on the serial interface and there is no way to avoid this).
  - ◇ Configuration of the WLAN module per RobotLoader software
- **Pre-installed bootloader,** therefore no ISP programming unit is required.
  - ◇ Programs can be uploaded via the RP6 USB interface into the microcontroller.
  - ◇ Programs can also be uploaded wireless via WLAN into the microcontroller at about the same speed as with a wired connection.
  - ◇ Numerous configuration options: Boot delay, LCD on/off, WLAN baud rate, deactivation of WLAN boot, adjustable boot vectors, automatic program start or start at the press of a button. A later extension of the bootloader with additional functions is possible (e.g. other protocols or other displays). This of course takes up additional flash memory space.
  - ◇ 8KB Flash ROM dedicated to the bootloader.
- **Dimensions (LxW):** 112mm x 90mm, numerous M3 mounting holes

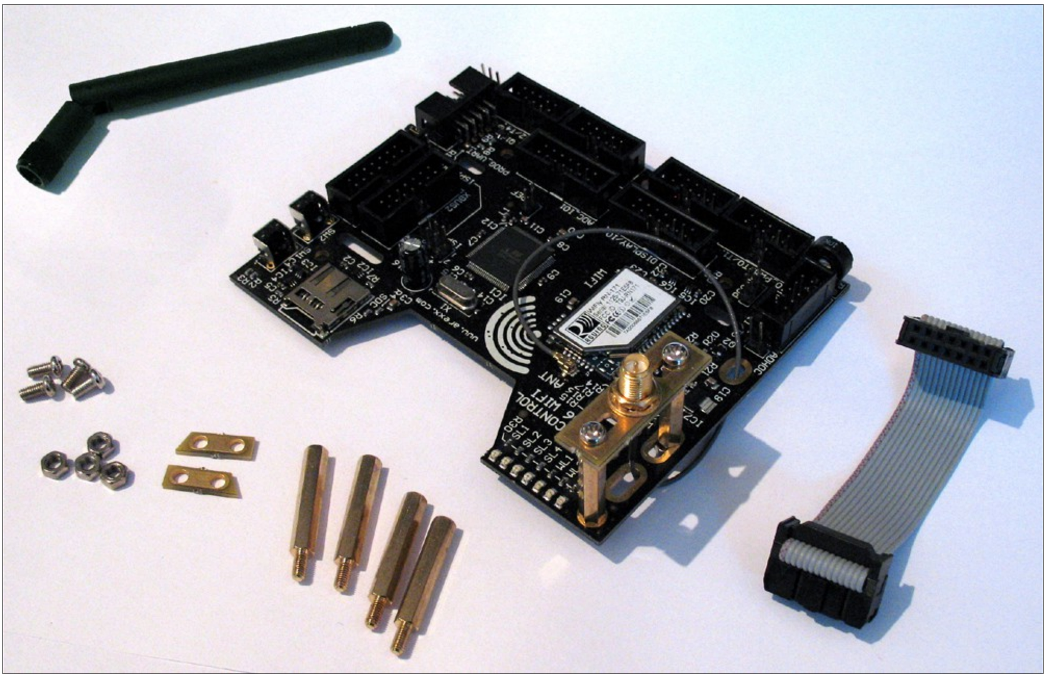
Like for the robot, some C example programs and a function library are included which makes starting with programming much easier.

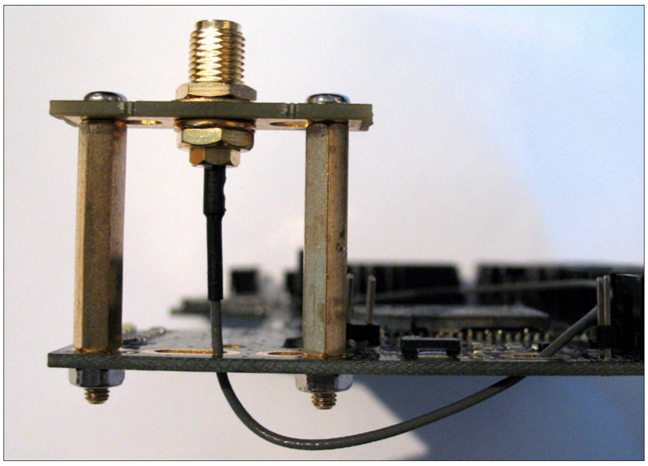
On the robot website, more programs and updates for the robot and its expansion programs may be available. Of course you can also exchange your own programs via the internet with other RP6 users! The RP6M256Library and the example programs are licensed under open source license GPL!

2. Installation of the Expansion Module

The way you can install the module on the robot depends on the other expansion modules that you might have already mounted on the robot.

The mounting hardware shown in the picture as well as antenna and connection cable should be included in the package:

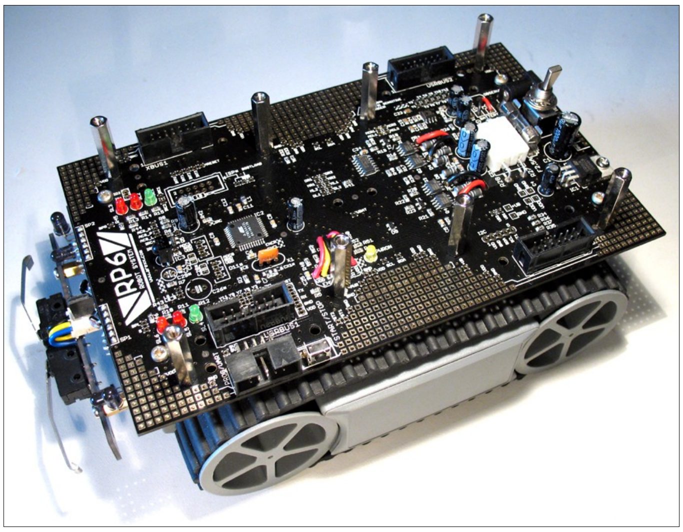




As you can see in the picture, the antenna mount with the gold-plated RP-SMA (Reverse Polarity SMA) connector **is already pre-installed.**

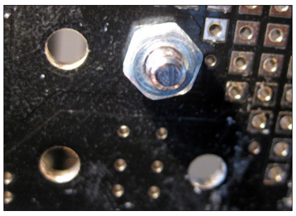
Please check if this is the case when you open the package. You should furthermore check if the antenna cable is passed through the PCB like on the picture. If this is not the case, you can correct it by yourself.

The RP-SMA connector should be firmly mounted on the antenna mount. If not, you can fasten it carefully via 2x pliers or 1x pliers and a 8mm hexkey (one plier holds the lower nut and the other (or the hexkey) turns the upper one). **Make sure not to slip off and damage the thread or the antenna cable!** The positioning of the flat part doesn't matter as you can rotate the antenna later at your convenience. The u.FL cable turns with the rotation which strains the small plug on the other side. Thus fastening via the flat side has been abandoned on purpose. This makes the mounting on other places easier (the connector can be rotated in such a way that the small u.Fl connector on the PCB is strained as little as possible).



Before you can mount the module on the robot, you need to loosen the four screws on the main board. Eventually you can also carefully loosen the small connector of the bumper PCB so that you can completely lift up the main board. However this is not necessary if you manage to pass your fingers under the main board in order to fasten the spacer bolts with the M3 nuts.

Caution: When you re-connect the cable to the bumper PCB, you have to pass one finger under the sensor PCB to compensate the pressure to avoid that the PCB is pushed too much backwards. Alternatively you can also unscrew the two screws on the bumper PCB and leave the cable mounted.

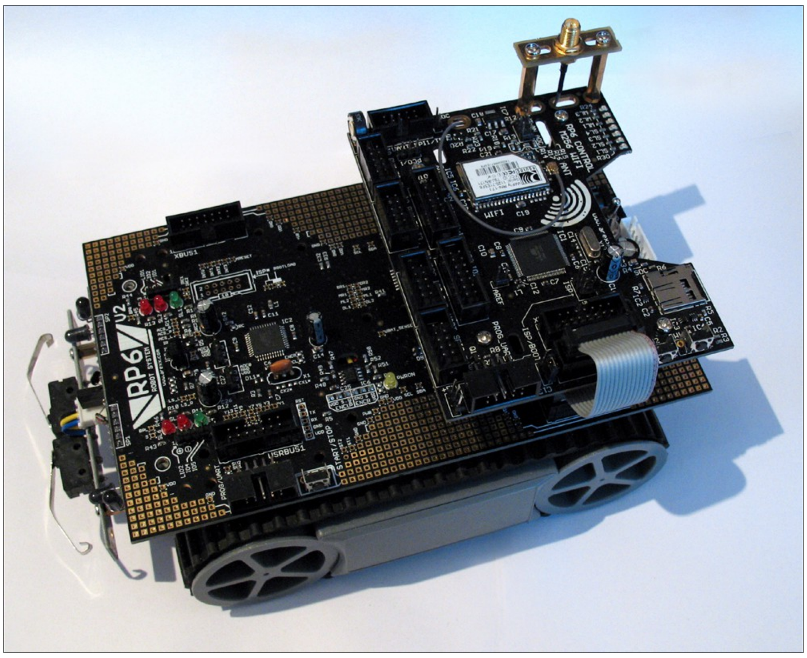


Then you can fasten the four 25mm M3 spacer bolts one after the other with the M3 nuts in the mounting holes on the main board as shown on the picture.

On the picture above, all 8 spacer bolts are fastened, also those of a second expansion module!

After that, you put the expansion module on top of the spacer bolts and fasten it via the four M3 screws. You can also proceed the other way round: start by screwing the spacer bolt on the expansion module and fasten it afterwards with the nuts on the main board.

Finally you have to plug in the 14-pin XBUS ribbon cable and that's it.

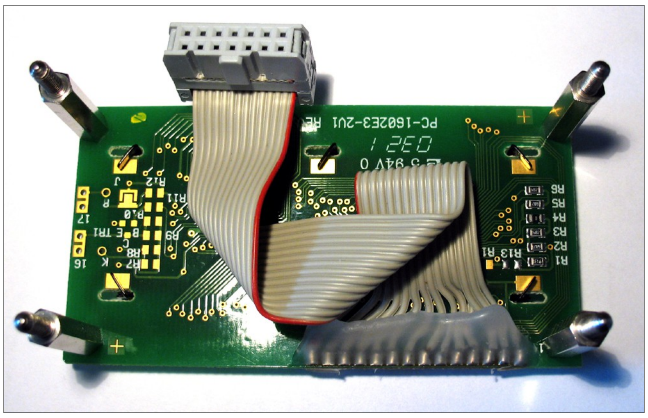


Unlike the other RP6 expansion modules, there is NO USRBUS connection because of a lack of space on the PCB at that location.

We recommend to mount the RP6-M256 on the rear expansion stack of the robot in the top position to optimize the installation of the antenna and the optional display. As a side effect, both programming connectors remain accessible on the same side of the robot. On the front edge you can mount the experiment expansion module RP6-EXP (see picture on the following pages for a configuration example).

If you also have the 16x2 characters LCD, you should connect and mount it to the expansion module PRIOR to installing it on the robot.

The 14-pin ribbon cable of the display is very flexible and can be easily folded. In order to fit well under the display, you should fold it for the RP6-M256 module like shown on the picture (may vary according to the display model!).

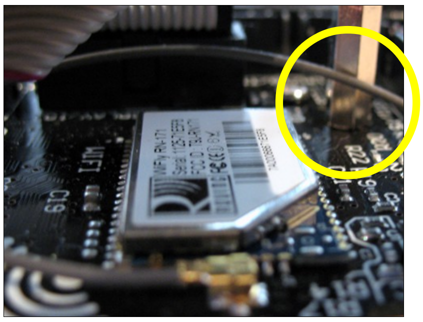


Now you can fasten the display e.g. with 20mm or 25mm spacer bolts, suitable nuts and screws on the expansion module.

You can also use any other text LCD with HD44780 compatible controller. You just need to solder a suitable cable to the display. *Please pay particular attention to the proper pin assignment! Note: The initialization carried out by the bootloader is set for a 16x2 characters display. The status messages can appear on larger displays as well but maybe in other places than expected.*

You don't need 4 spacer bolts as shown on the picture. 2 pieces (<u>both</u> on the front *or* on the rear) are sufficient to fasten the display.

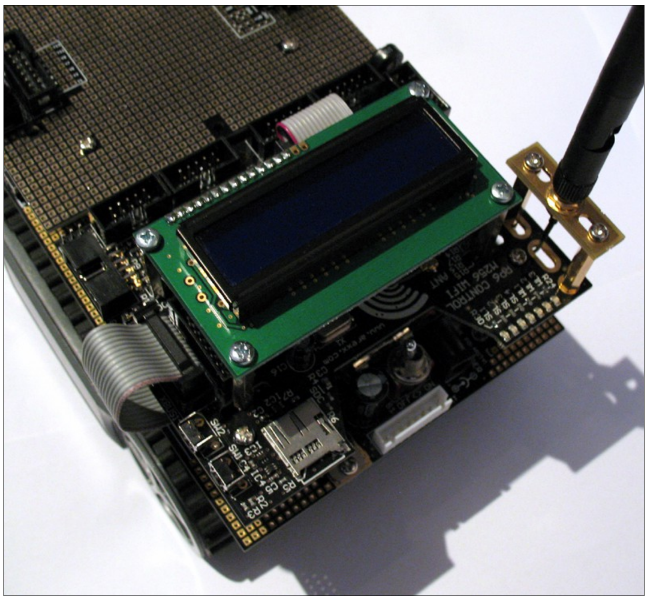
CAUTION: During installation of the display, please make sure that the thin antenna cable is not clamped <u>under</u> the spacer bolt! This can happen easily, so push the cable carefully aside when you fasten the spacer bolt!



Tip:

The RP6-M256 is supplied with two small PCB pieces with two M3 drill holes. These are OPTIONAL (i.e. they are normally not necessary) and can be used as adapters for LCDs where the mounting holes don't match exactly to the drill holes on the PCB. You just might need to fit the height of the spacer bolts with nuts or rings.

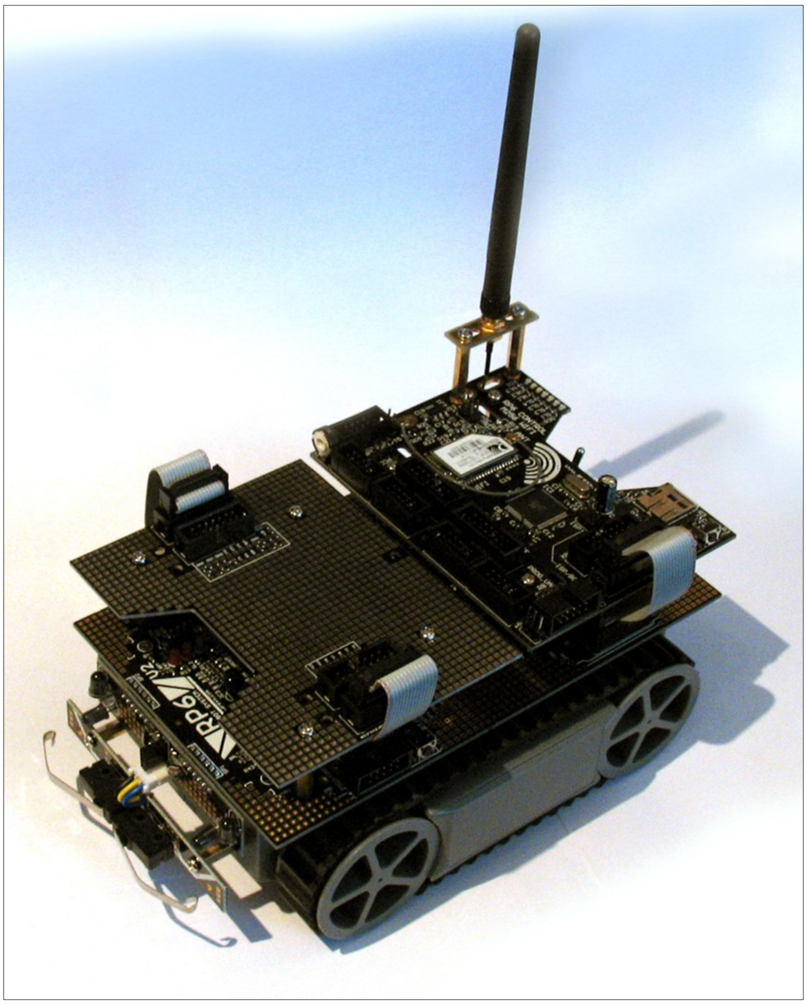
The PCB pieces can also be used for other mounting purposes, e.g. to install the antenna mount at another location (with slightly inappropriate drill holes). You eventually only need two additional M3 screws and nuts.



When it is mounted, the display could look like on the picture on the right.

The expansion connectors with the free I/O ports can simply be connected via small 10- or 14-pin flat ribbon cables to one or more RP6-EXP expansion modules. The I/Os and ADCs can be used there for reading sensors, etc. One single module will hardly be sufficient to use the numerous I/O ports in a sensible manner. It would be a more flexible solution to mount two RP6-EXP modules next to each other on the first level and install RP6-EXP on the front and the RP6-M256 at the rear side on a second level.

A combination with the RP6-M32 module would also be possible. A combination with the RP6-M128 does not make so much sense as unfortunately the C-Control software doesn't support the I2C slave mode. The RP6-M256 module should better remain the bus master. An alternative would be to drive the C-Control module via one of the secondary serial interfaces. This of course requires several software adjustments.



The ribbon cables fit perfectly through the center gap between the two module stacks.

Advanced user can route the wire of the WLAN antenna through there as well in order to mount the antenna on another level or on a self-made construction.

The RP6-M256 module can be used in various totally different applications. In that case, the flexible antenna mount is an advantage (e.g. installation in a case).

3. RobotLoader 2

The new RobotLoader version 2.x (up to version 1.4 called RP6Loader) offers some new functions, especially support for network connections via the WLAN module. The new features will be briefly explained in this section.

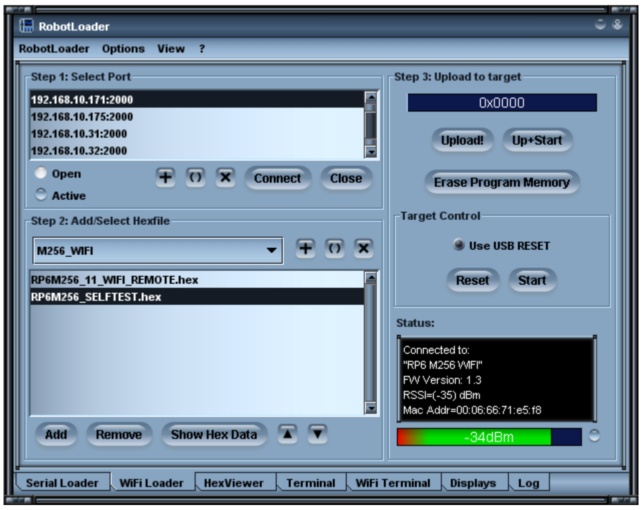
3.1. Summary

Exactly as for the RP6Loader and RobotLoader 1.x, the RobotLoader 2 doesn't need to be installed. Just unzip it to any folder on the hard disk and launch the file RobotLoader_start.exe or for Linux use the startscript robotloader_linux.sh or robotloader_linux_x64.sh (for 64 Bit systems).

As shown in the figure below, you can load software into the microcontroller via the new WiFi loader just as you would do via a serial USB connection. The upper part shows a list of IP addresses in <IP>:<PORT> format. With the small buttons

+ *add*, () *change* and **x *erase***

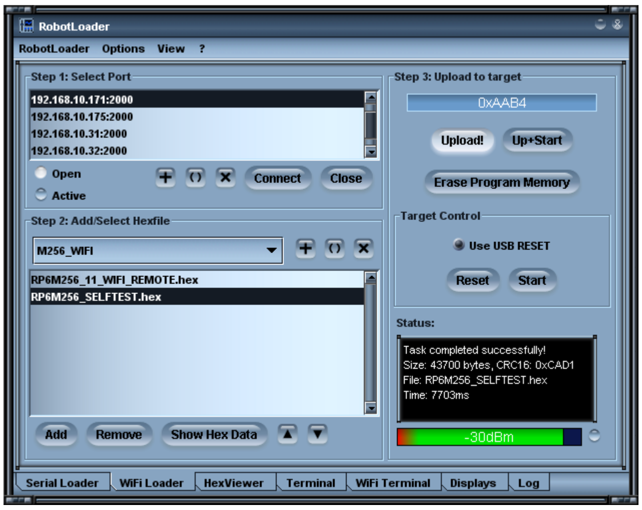
you can edit the list manually exactly like the hexfile list. The input of IP and port must also be done in <IP>:<PORT> format e.g.: 192.168.10.171:2000. The standard communication port is 2000. This can however be changed in the settings of the WLAN module.



192.168.10.171:2000
192.168.10.175:2000
192.168.10.31:2000
192.168.10.32:2000

RP6M256_11_WIFI_REMOTE.hex
RP6M256_SELFTEST.hex

```
Connected to:
"RP6 M256 WIFI"
FW Version: 1.3
RSSI=(-35) dBm
Mac Addr=00:06:66:71:e5:f8
```



192.168.10.171:2000
192.168.10.175:2000
192.168.10.31:2000
192.168.10.32:2000

RP6M256_11_WIFI_REMOTE.hex
RP6M256_SELFTEST.hex

```
Task completed successfully!
Size: 43700 bytes, CRC16: 0xCAD1
File: RP6M256_SELFTEST.hex
Time: 7703ms
```

Basically you just have to select the desired address, select the hexfile (exactly as in the normal “serial loader” tab) and click on “Upload”. The speed is roughly the same when the connection quality is good and does not have high latency.

When you click on Connect, communication is established and status information is requested from the RP6-M256. This is shown in the lower area in the “Status” box. Just below is the RSSI bar-graph (Received Signal Strength Indicator) that shows the currently received signal strength in dBm (deciBel with a reference value of 1mW, e.g. 0dBm = 1mW, -30dBm = 0.001mW). The bar-graph is updated at every click on “Connect” and also via a UDP broadcast that is sent every few seconds (adjustable) from the WLAN module. As long as this bar is still green on the right side, the signal strength is OK (everything better than -65dBm). If the bar becomes completely red, the signal is weak and the communication might be interrupted if the distance between the robot and the access point keeps increasing.

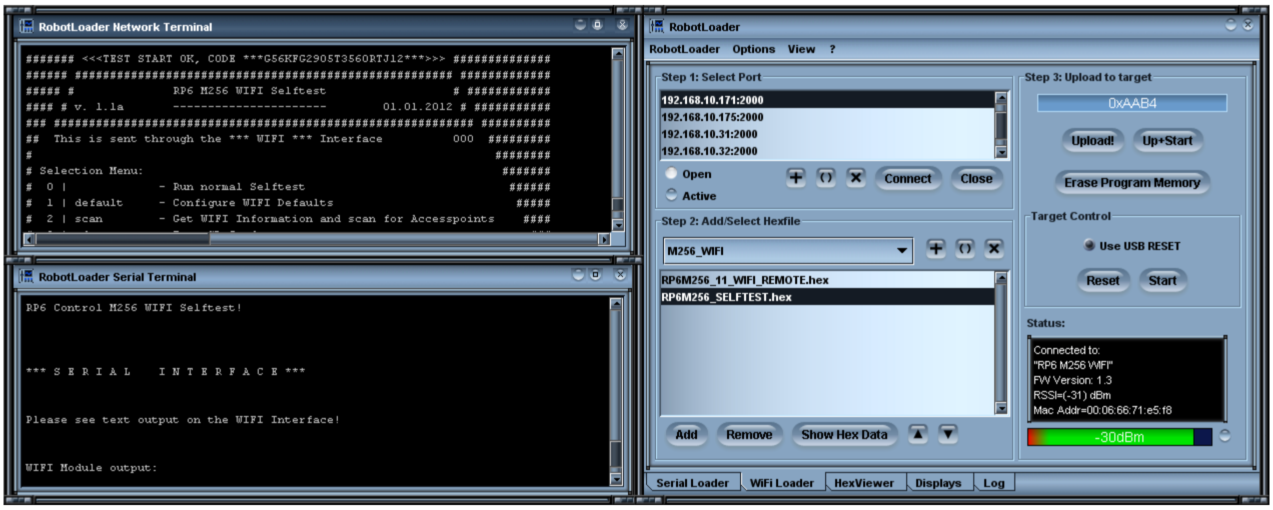
Acceptable:

Bad:

The transmission range of the WLAN is strongly dependent on the environment. Any kind of obstacle will more or less reduce the range (especially water and metal). Nearby WLAN networks, 2.4GHz video transmitters and microwave ovens can also interfere with the own WLAN.

3.2. New Terminals

The serial terminal alone is not sufficient anymore and thus a network / WIFI terminal has been added. Basically it behaves almost like the normal serial terminal. As it will be necessary to switch more often between the tabs, it is possible to detach the terminals into separate windows that can be freely positioned. So the screen could look like the example below:



```
####### <<<TEST START OK, CODE ***G56KFG2905T3560RTJ12***>>> ##############
###### ################################################## #############
##### #              RP6 M256 WIFI Selftest                  # ############
#### # v. 1.1a       ----------------------         01.01.2012 # ###########
### ##################################################### ##########
##  This is sent through the *** WIFI *** Interface          000  #########
#                                                                ########
# Selection Menu:                                                 #######
#  0 |              - Run normal Selftest                          ######
#  1 | default      - Configure WIFI Defaults                       #####
#  2 | scan         - Get WIFI Information and scan for Accesspoints   ####
```

```
RP6 Control M256 WIFI Selftest!



*** S E R I A L   I N T E R F A C E ***


Please see text output on the WIFI Interface!


WIFI Module output:
```

Step 1: Select Port
192.168.10.171:2000
192.168.10.175:2000
192.168.10.31:2000
192.168.10.32:2000
Open
Active

Step 2: Add/Select Hexfile
M256_WIFI
RP6M256_11_WIFI_REMOTE.hex
RP6M256_SELFTEST.hex

Step 3: Upload to target
0xAAB4

Target Control
Use USB RESET

Status:

```
Connected to:
"RP6 M256 WIFI"
FW Version: 1.3
RSSI=(-31) dBm
Mac Addr=00:06:66:71:e5:f8
```

(The window size is adjustable)

Now the serial interface and the WIFI terminal can be used simultaneously for outputs of the program which can be very useful during the development stage of new software for the RP6 e.g. display at the same time outputs from the controller on the main board via the serial interface and outputs of the RP6-M256 via WLAN, or generate two different outputs from the RP6-M256.

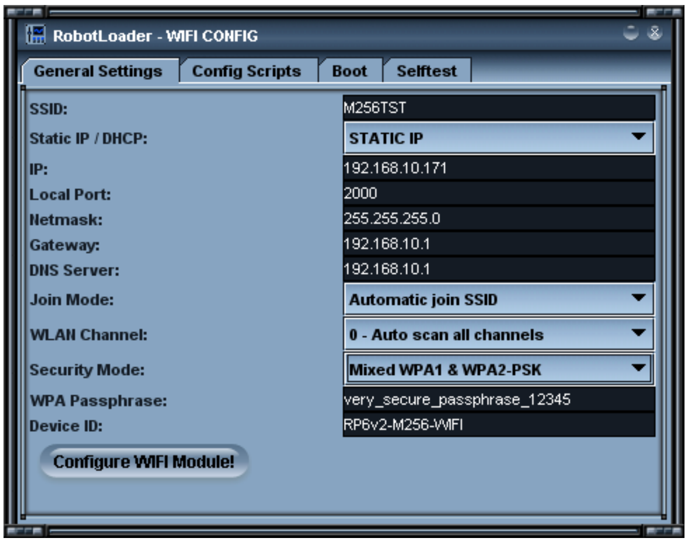
Tip:

It is also possible to use other terminal software like PuTTY or Tera Term (both should have faster display refresh rate at high transmission rates). If they are set to Telnet, a communication with the WLAN module should be possible. Maybe you need to adjust the line breaks (CR or CR+LF is used by the WLAN module interpreter, LF or CR+LF by the example programs, so select CR+LF)

Unlike with the serial connection, the WLAN connection doesn't allow you to start the loaded program by a simple "s" but with the command "start_program" instead (this is more reliable via the WLAN connection, also because of various status messages of the WLAN module).

3.3. WLAN Settings

The WLAN module can be set up via a configuration dialog. To this end, the USB serial interface must be connected to the RP6-M256 module. The settings are automatically stored in the WLAN module and remain there even when the module has been switched off.



SSID:	M256TST
Static IP / DHCP:	STATIC IP
IP:	192.168.10.171
Local Port:	2000
Netmask:	255.255.255.0
Gateway:	192.168.10.1
DNS Server:	192.168.10.1
Join Mode:	Automatic join SSID
WLAN Channel:	0 - Auto scan all channels
Security Mode:	Mixed WPA1 & WPA2-PSK
WPA Passphrase:	very_secure_passphrase_12345
Device ID:	RP6v2-M256-WIFI

Currently the dialog supports changes of the settings only if the USB cable is connected. Via the network terminal, the settings can also be configured via a command-line interpreter that is integrated in the WLAN module. The manufacturer of the module offers a comprehensive documentation on the subject. For simple applications, it is sufficient to configure the module one time without the need of changing the settings again later on. From the point of view of the microcontroller, the module behaves like a serial interface – with the difference that a normal TCP/IP connection can be used on the PC side. This can be easily used with any current programming language, and even (restricted) access per web browser is possible.

All usual parameters of the used WLAN access point must be provided in the dialog. At the end, just click on "Configure WIFI Module" (prior to that select the appropriate port in the serial loader tab). Then the RobotLoader switches automatically to the "Log" view and displays the commands that have been executed and if they were successful (green "OK!" messages should appear after the commands!). You can monitor this on the serial terminal as well. Everything that happens there can also be done manually via text commands but the user interface makes this much easier.

In the "Configure Scripts" tab of the configuration dialog, you can use configuration scripts which are simple text files (some examples included in the RobotLoader directory). This is useful if you have to change the WLAN access point often, e.g. if the robot is used in different locations.

After this general presentation, we will now explain the configuration step by step.

WARNING:

There are several advanced settings in the "Boot" Tab of the WLAN configuration dialog, you can also change the baudrate of the WLAN module and bootloader there. **DO NOT change the Baudrate without any good reason! You may have to do a manual factory reset using the ADHOC Jumper if something goes wrong here...** (s. troubleshooting section in this manual)

In the appendix you will find many useful tips if you experience any problems!

3.3.1. Configure Access Point

The configuration of the WLAN access points varies from one model to another. We recommend to read the accompanying documentation. If possible you should use fixed IP addresses (DHCP deactivated) which makes it easier to address the WLAN module. The RobotLoader software can find the WLAN module automatically via the dialog box *Options->Discover WIFI Devices*. There, the UDP status messages sent by the WLAN module are displayed as a list. Any number of RP6 WLAN modules can be listed here. Click on the "Add" button to add the IP to the list in the WIFI loader tab.

This is particularly useful if you use DHCP. For own applications, it is easier though to use static IP addresses.

WPA is mandatory as encryption mode! Do <u>not</u> use WEP! This encryption mode has been cracked a long time ago and is no serious obstacle anymore. Of course don't use the standard settings of the router as a password but find a good one by yourself (by the way, the password shown in the figure is not a very good example ;-)).

Choose the other settings according to your network. You can try out various WLAN channels if some are strongly disturbed by neighbor networks or other sources of interference. Channel 1, 6 or 11 are not overlapping in the 2.4GHz band.

Tip:

On a Notebook with WLAN you can use e.g.. inSSIDer

http://www.metageek.net/products/inssider/

(for Linux available as a BETA version) to display the channel assignment and select a relatively free channel.

Another tip:

If the access point is only used for the robot control, you can also use the slower 802.11b mode. The advantage is that the range is slightly increased. However you need to manually change the configuration of the RN-171 module. The access point can also be set to b/g (or b/g/n) mixed mode. However this mode reduces the transmission speed of units that work in the faster 802.11g/n mode.

In the WLAN command-line (or in the configuration script)

```
set wlan rate 8
```

activates the normal 6MBit/s 802.11g mode (6MBit/s is the slowest rate in 802.11g mode, allows higher transmit range) and

```
set wlan rate 1
```

switches to 2MBit/s 802.11b. You can not use that mode if you have configured your router to 802.11g only!

Higher quality dual band routers featuring two HF modules allow to assign the faster devices to a separate channel.

3.3.2. PC configuration

The computer on which you use RobotLoader must be configured according to the access point / router settings. It is recommended to use static IP addresses. Configure subnet masks and eventually the gateway accordingly (router IP; this must not necessarily be the access point but may also be a second separate router for the internet connection). It is impossible to explain this for all operating systems and network configurations in this manual but it is generally very well documented.

It can look similar to the figure of the WLAN configuration dialog, e.g. netmask 255.255.255.0, IP 192.168.10.x (x = 1 to 254, but not 255, because that would be the broadcast address!), ...

If possible, test the WLAN connection with another device before you try it with the RP6-M256. Notebooks, tablets or smartphones with WLAN function are perfect for this purpose. If it works with one of these, you can setup the RP6-M256 accordingly.

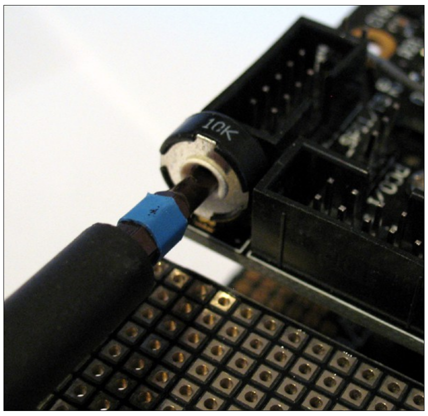
3.3.3. Test of RP6-M256 USB Connection and Display

Now you can test and configure the RP6-M256.

First check if the ISP / BOOT jumper (small 3 pin connector between the XBUS and PROG / UART connectors) is set to position BOOT (i.e. the position at the edge of the PCB). If this is not the case, it will not work because the reset signal can't be triggered.

Now connect the USB interface on the PC to the PROG/UART connector on the RP6-M256 via the 10-pin ribbon cable and start the RobotLoader. Turn the robot on. If you have an LCD, a text message should be displayed. The status LEDs should flash (WL1-3 are driven by the WLAN module and SL1-4 by the microcontroller. This means that several red, green and yellow LEDs should flash several times just after power on). If this worked, click on "Connect" in the Serial Loader tab of the RobotLoader – a message like "Connected to RP6 Control" should appear in the status box.

Please note: Do *not* mix up the SERIAL loader used for this test, with the WIFI Loader! At this stage it can't work with the WIFI Loader as the WLAN module hasn't been configured, yet.



If the display is blank or shows only two lines of completely dark boxes but the LEDs are flashing and the connection with the RobotLoader works, you must set the contrast of the display (or you have used a different display and the pin assignment is wrong.... Switch the robot off immediately and check the connection). Set the contrast with potentiometer R1 on the PCB via a small flat head screwdriver. Cross-head works as well but might be more difficult as they don't fit properly into the potentiometer.

Remove the ribbon cable of the experiment module (if installed) to get better access to the potentiometer.

3.3.4. Configuration and Test of the WLAN Module

Once the first test has been done, you can immediately start with the WLAN configuration dialog: Menu `Options → WiFi SERIAL Config → Configure WiFi Settings`.

Prior to that, switch to the WIFI Loader tab in the main window of Robot-Loader and make sure that the checkbox `Use USB Reset` is enabled. This is important as the WLAN module can't reset the AVR microcontroller if the USB interface is connected. This is also necessary if the USB interface is connected to another microcontroller (common reset signal, the USB interface always keeps the control when it is connected).

Re-check all settings once again in the configuration dialog and click on the "Configure WIFI Module" button. The rest runs as described above. *If problems arise at this stage,* the WLAN module might still use the default factory settings. In this case, you have to run an initialization script in the "Configure Scripts" tab. Click the button

`Run Initial WIFI config (BAUDRATE IS 9600).`

If this remains unsuccessful, check the power supply and if the WLAN LEDs really do flash (one of the yellow LEDs and the green one should flash: this means that it tries to establish a connection to the preset WLAN which will of course fail). In case of problems please contact the customer support.

If everything went well and the WLAN access point is switched on and within range, none of the yellow LEDs should be on anymore and only the green LED should be flashing slowly. This means that the WLAN module has set up a connection.

Detach the network terminal via the menu `View->Detach Network Terminal` and place it next to the RobotLoader window. This eases the operation and the communication with the WLAN module is more convenient to follow.

Now you can test the connection with RobotLoader. For this purpose add the correct IP address to the list in the WiFi Loader tab of RobotLoader e.g.

`192.168.10.171:2000`

Click on "Connect" - the status box should now show information about the module (MAC address, firmware version) and display the signal strength. Some text outputs appear in the terminal. The green LED stops flashing and remains steadily on (open connection).

Please note:

If "Timeout" or similar error messages appear and no connection can be established after further automatic trials, just click again on Connect. This should not happen very often but may occur depending on the network. In case of connection problems, first click on "CLOSE" and try again a few times.

If this worked, you can load the self-test program via the WLAN connection into the microcontroller and start it. Please use the WLAN connection in order to test if everything works, i.e. do not use the SERIAL Loader tab in this case!

After a successful upload click on the "Start" button.

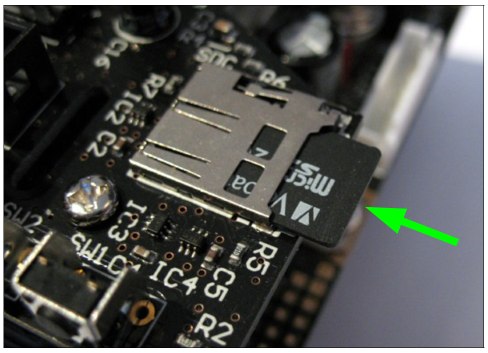
Does the program run properly? A text menu should be displayed in the network terminal. Please note that this menu does NOT appear in the normal serial terminal, so don't mix up the two different terminals!

3.3.5. MicroSD Card

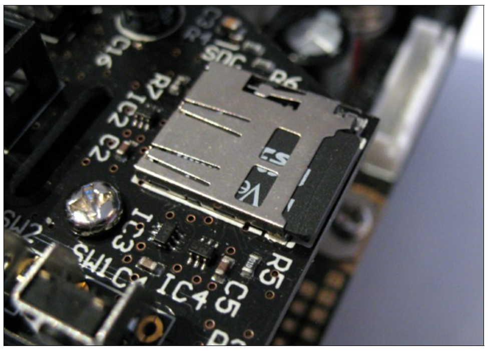
You need a microSD card to finish the self-test completely. The SD card must be formatted with FAT32. NTFS or Linux ext3 are not supported. If you don't have a microSD card, you can skip this step. *An **error message will show up** at the end of the Test because no SD card has been found. You can of course ignore that message.*

There is NO WARRANTY for potential data losses! The card is directly accessed without any operating system. The card can be written directly without regarding the the file system – which may result in a corrupted file system. Moreover programming mistakes might easily lead to data losses. So please use a card that doesn't contain any important data! MicroSD cards are very cheap and it shouldn't be a problem to have a small 1 or 2GB card dedicated only to the RP6-M256 module.

To perform the SD card test, you have to copy a test file from the PC into the main directory of the SD card via an appropriate card reader. You will find the file in the directory of the selftest program. It is called “M256_SELFTEST_TESTFILE.txt” and contains some ASCII text that is read by the program and sent to the PC via the WLAN connection. A test string at the beginning of the file will be checked and only if the text has been read properly, the test is passed successfully.



Insert the SD card as shown into the card slot. This is a so-called push-push slot, i.e. you have to push the card slightly to insert it and push once again and then pull to remove it. You will hear a little click sound. For safety reasons please turn the robot off BEFORE and turn it on only AFTER having inserted the card!



CAUTION:

The microSD card slot is NOT intended for changing a card during operation! It might work but we can't guarantee for anything!

In general there is no warranty for any data losses! It is highly recommended to use an empty card.

Now you need to load the RP6Base_I2CSlave.hex program into the microcontroller on the RP6 main board! This is necessary to have a suitable I2C bus partner available with whom the bus communication can be tested. Once the program has been loaded to the MEGA32 on the RP6Base, please switch the connection of the USB interface back to the RP6-M256.

Then you can start the selftest. As soon as the selection menu is displayed, type in a 0 and press Enter. Outputs will now be sent on the serial interface and via the network connection. You should also see outputs on the LCD (if one is connected). Please follow the instructions of the program.

First of all, press the two buttons SW1 and SW2. Just afterwards the red LEDs will be tested – so please pay attention if they all light up (only the red LEDs are directly controlled by the microcontroller, the other LEDs are linked to the WLAN module)!

The I²C test performs a small running light with the LEDs on the RP6 main board to check the communication.

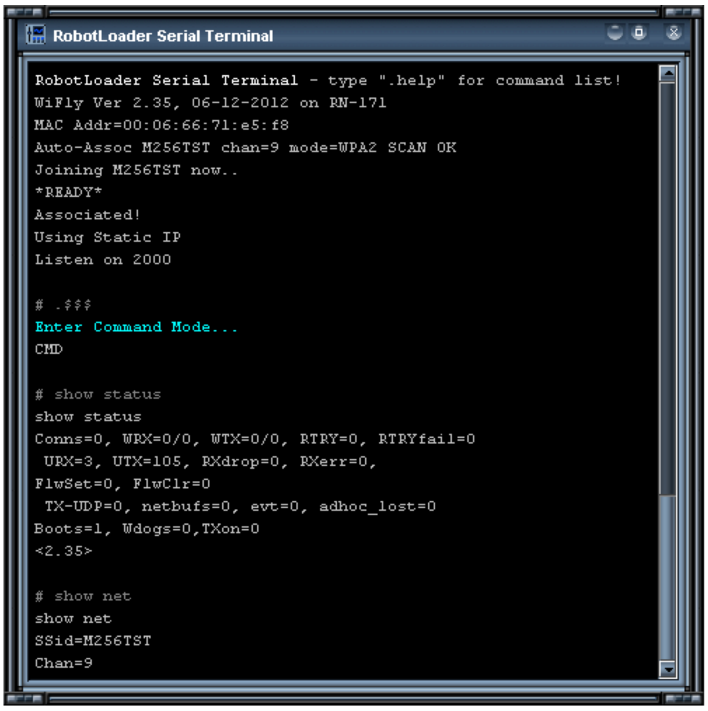
At the end of the test, the text is read and output from the SD card. **If no SD card had been inserted, you will get an error message. In this case, you can of course ignore it.** It will show a test failed message but the module will usually be just OK, you can test it again later when you have a suitable card.

If all this is working perfectly, you can start with the example program once you have finished to read the manual!

3.4. WLAN Command-Line Interface

There are 2 ways to access the WLAN module via the terminals. One is of course via the network which is however only possible if the connection is working properly. The other way is via the serial interface (USB interface). For this purpose, the microcontroller switches into a "Passthrough" mode where it transfers everything from the PC to the WLAN module and the other way round. Therefore the WLAN module is very convenient to configure. The microcontroller doesn't need to set anything on its own in the program as the settings of the WLAN module can be stored in internal memory.

To activate the passthrough mode, just click on `Options → WiFi SERIAL Config → Enable Passthrough Mode` in the RobotLoader menu. After that, all inputs into the serial terminal are transmitted to the WLAN module (please make sure that CR or CR+LF are activated in the options!).



```
RobotLoader Serial Terminal - type ".help" for command list!
WiFly Ver 2.35, 06-12-2012 on RN-171
MAC Addr=00:06:66:71:e5:f8
Auto-Assoc M256TST chan=9 mode=WPA2 SCAN OK
Joining M256TST now..
*READY*
Associated!
Using Static IP
Listen on 2000

# .$$$
Enter Command Mode...
CMD

# show status
show status
Conns=0, WRX=0/0, WTX=0/0, RTRY=0, RTRYfail=0
 URX=3, UTX=105, RXdrop=0, RXerr=0,
FlwSet=0, FlwClr=0
 TX-UDP=0, netbufs=0, evt=0, adhoc_lost=0
Boots=1, Wdogs=0,TXon=0
<2.35>

# show net
show net
SSid=M256TST
Chan=9
```

If you type in:

```
.$$$
```

you will enter the command mode (the dot at the beginning is only necessary in the RobotLoader terminal and makes sure that a delay is inserted before and after and that NO CR+LF is sent – it is not easily possible to do it differently in the RobotLoader terminal as the transmission starts only when the enter button is pressed and then the separation sign is added automatically).

The command mode offers many options to configure the WLAN module and to request the current status. The table on the next page shows a few useful commands but it is far from being exhaustive. Please refer to the detailed documentation of the manufacturer.

`ver`	Display the firmware version.
`show rssi`	Display the current signal strength.
`scan`	Scan for access points
`show status`	Display current status
`show net`	Display current network status
`get everything`	Display all settings
`get wlan`	Display WLAN settings
`get ip`	Display IP settings
`get comm`	Display communication parameters
`set ip adr 192.168.10.171`	Set IP address
`set ip mask 255.255.255.0`	Set network mask
`set ip gateway 192.168.10.1`	Set gateway
`set ip dhcp 0`	Deactivate DHCP
`set wlan join 1`	Connection mode to access point
`set wlan ssid M256TST`	Use access point with SSID M256TST
`set wlan auth 4`	Use WPA2 access points with automatic connection to the next access point
`set wlan passphrase <PASS>`	Set WPA passphrase
`set wlan hide 1`	Hide WPA passphrase (for get command)
`set wlan linkmon 5`	Send every 5 seconds UDP broadcast on port 55555 and check connection to the access point.
`set comm size 1024`	Set network packet size. Here: if 1024 characters are in the buffer, send network packet.
`set comm time 10`	Set Timeout for the transmission of the network packet. Here: if no new data have been received for 10ms via the serial interface, send network packet.
`save`	Save settings in the flash ROM
`exit`	Quit command mode
`reboot`	Re-start module

As already said, the table is far from being complete and gives just a quick idea. You don't have to do all this by yourself as the RobotLoader can make these settings for you.

Please don't change the I/O port parameters without a good reason because they have been set to allow resetting of the microcontroller via the WLAN module which is absolutely necessary for the bootloader.

The Passthrough mode can also be activated via the two push-buttons on the module. If necessary, you can also switch into the 9600 baud mode if the default settings of the WLAN mode have been reset (switch off the robot, keep SW1 pressed, switch the robot on → Passthrough mode 9600 baud is active, module can be configured. Switch the robot on and wait until the bootloader timeout is passed, then press SW2 → 500k-Baud Passthrough mode is now active).

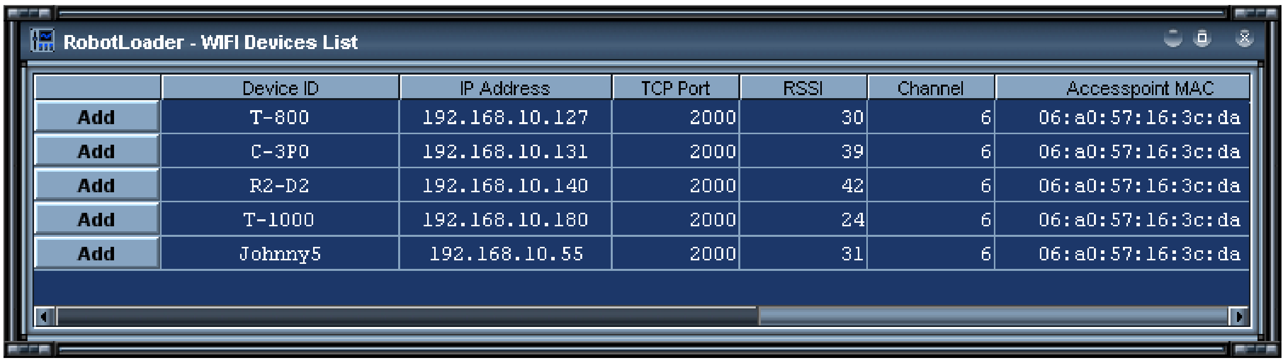
Normally you don't have to do this because the RobotLoader can also activate these modes without the press on a real physical button of the module.

You can also enter command mode via the network connection. It works quite similar via the input of .$$$

The commands are identical but please note that the connection is closed when you change the settings and restart the module.

3.5. Find out the WLAN IP Address

If the IP address of the module is unknown or is dynamically assigned per DHCP, you can find it in the "Discover WIFI Devices" dialog.


RobotLoader - WIFI Devices List

	Device ID	IP Address	TCP Port	RSSI	Channel	Accesspoint MAC
Add	T-800	192.168.10.127	2000	30	6	06:a0:57:16:3c:da
Add	C-3PO	192.168.10.131	2000	39	6	06:a0:57:16:3c:da
Add	R2-D2	192.168.10.140	2000	42	6	06:a0:57:16:3c:da
Add	T-1000	192.168.10.180	2000	24	6	06:a0:57:16:3c:da
Add	Johnny5	192.168.10.55	2000	31	6	06:a0:57:16:3c:da

The WLAN module can send a status message every few seconds per UDP broadcast to port 55555. New IP addresses will appear in the list and the log of RobotLoader. By the way, the displayed device ID can be changed in the WLAN settings. However don't use spaces as this is a control character of the command interpreter!

After powering the unit on, it can take up to 10 seconds before a status message arrives. It doesn't need to establish a connection for this. However check the firewall settings before in order to allow UDP data on port 55555.

Click on the "Add" button to add the IP to the list in the WiFi Loader tab.

3.6. Solving WLAN Connection Problems

If you experience problems with the connection to the WLAN module, this can have various reasons. Some of these are briefly discussed here.

The most important thing: The WLAN access point must be correctly configured! If the WLAN doesn't work with other devices, it will probably not work with the RP6-M256 as well. The settings of the PC and of the WLAN module must equally match. If this is not the case, the connection can't work.

If you are sure that the WLAN works correctly, you need to find out what goes wrong with the communication to the RP6-M256.

As described above, you can switch the microcontroller into “passthrough” mode in RobotLoader and send commands directly to the WLAN module. Then you have various options for trouble-shooting (see Appendix B for further details). Check if the WLAN module can connect to the access point (the yellow WL1 LED will flash if no connection can be established with the access point). If the WLAN connection works, the problem might come from the PC settings.

Check if the communication is blocked by a firewall! Windows PCs normally feature at least the Microsoft firewall. At the first start of the new RobotLoader, a security control dialog might appear as the RobotLoader wants to listen on UDP port 55555 immediately after starting. That should be allowed! Of course TCP data exchange with the WLAN module must be allowed as well, otherwise it will not work at all.

Please make also sure that the WLAN module uses an IP address that is used by no other computer in the LAN. All computers must be in the same IP sub-network i.e. with netmask 255.255.255.0 the first three numbers of the IP address must be the same. This means that all IP addresses would start like this: 192.168.10.x and the last number instead of the x can be a number between 1 and 254. If the computers are in the subnetwork 192.168.2.x or anything similar, it is impossible to communicate directly with the WLAN module if its IP is IP 192.168.10.x.

If you want to use DHCP, make sure that the DHCP server works correctly and the DHCP mode is activated in the WLAN module.

In the appendices you will find many tips that will help you with WLAN problems.

If you didn't find your specific problem, here once again the tip from the start:

In case of problems you can ask your questions quickly and easily per e-mail:

`info@globalspecialties.com`

You will find documentation software updates and further information on the robot homepage:

www.globalspecialties.com

4. RP6 CONTROL M256 WIFI Library

As for the robot itself, there is a comprehensive function library for the RP6-M256 with many useful features that make life easier for beginners. The name of the library is RP6 CONTROL M256 WIFI Library or a bit shorter: RP6M256Lib. The stopwatch, delay, UART and I²C-bus functions are identical to those of the normal RP6Library. The I²C bus even uses the same source code files. These are stored in the sub-folder RP6common of the RP6Library. The UART library is different as the internal structure of the ATMEGA2560 is a bit different and the UART routines had to be optimized for the WLAN module. We will not describe these functions again as they are mostly identical. **Refer to the relevant chapter in the RP6 manual an the example programs!** Here we only describe the functions that are exclusive to the RP6-M256 or that work a bit differently in the RP6Lib.

Tip:

Despite all the ready to use functions, the RP6M256Lib is only a starting point and far from being perfect. You can improve it and add many additional things. Here you have to use your own programming skills!

New versions of the library will be published online. This applies also to the example programs.

4.1.1. Initialization of the Microcontroller

```
void initRP6M256(void)
```

As you already know from the RP6Lib, this function must ALWAYS be called first in the main function! It just has a bit different name here.

The function initializes the hardware modules of the microcontroller on the RP6-M256. Only if you call this function right at the beginning, the microcontroller will work properly! A part will be initialized by the bootloader but not everything.

Example:

```
1  #include "RP6M256Lib.h"
2
3  int main(void)
4  {
5          initRP6M256();              // Initialization
6                                      // ALWAYS CALL THIS FIRST!
7  // [...] Program code...
8
9          while(true);       // Infinite loop
10         return 0;
11 }
```

Every program for the RP6 CONTROL M256 must <u>at least</u> look like this! The infinite loop in line 9 is necessary to ensure a defined end of the program!

Without this loop, the program could behave differently than expected. *Exactly as for the controller on the main board!*

Usually something useful is done in this loop. Ideas for the general structure of programs can be found in the example programs!

4.1.2. Status LEDs

The LEDs are controlled like the ones on the main board, however only four LEDs are available.

This function is called the same as on the main board:

```
void setLEDs(uint8_t leds)
```

Example:

```
setLEDs(0b0000); // This command turns off all LEDs.
setLEDs(0b0001); // This one switches LED1 on and all others off.
setLEDs(0b0010); // LED2
setLEDs(0b0100); // LED3
setLEDs(0b1010); // LED4 and LED2
```

The LEDs are directly linked to the I/O ports of the microcontroller (unlike the RP6-M32 where these and the LCD are connected to an external shift register). There are also 4 functions that set the LEDs individually:

```
void setLED1(uint8_t led),  void setLED2(uint8_t led),
void setLED3(uint8_t led),  void setLED4(uint8_t led)
```

```
setLED1(1); // LED1 on
setLED2(0); // LED2 off
```

4.1.3. Buttons

The two buttons on the board can be queried via the following routines.

```
uint8_t getPressedKeyNumber(void)
```

This functions finds out which button is pressed and returns 1 or 2. If no button is pressed when it is called, it returns 0.

```
uint8_t checkPressedKeyEvent(void)
```

This function checks if a button is pressed and returns one single time the button number - unlike getPressedKeyNumber, where the button number is sent back permanently. This is useful to query the buttons in the main loop without interrupting the program flow.

Following function works in a similar way:

```
uint8_t checkReleasedKeyEvent(void)
```

It returns the button value only once and only when the button has been pressed and also released. This function doesn't block the normal program flow, so you don't have to wait in a loop until the button is released.

4.1.4. LC-Display

The LCD is ideal to display simple sensor values and status messages while the robot is not connected to the PC. The output on the LCD works in a similar way as with a serial interface – however there are a few peculiarities. Just look at the example programs and you will understand quite quickly how the LCD can be used.

`void initLCD(void)`

This function must always be called at the beginning of the program to initialize the LCD. If you connect other peripherals to the I/O ports, you should remove these and other LCD functions from the relevant programs.

`void setLCDD(uint8_t lcdd)`

Normally you don't need this function. We only mention it here to show how the display is controlled.

The LCD can be operated in 8-bit mode. 8 data lines and 3 control lines are used (Enable (EN) and Register Select (RS), Read/Write (R/W), which means that all signals are used, unlike on the RP6-M32 and RP6-M128! This is not so important for text displays but might well be for larger dot matrix displays for performance reasons). This function sets also the Enable signal so that the LCD accepts the data.

`void writeLCDCommand(uint8_t cmd)`

This function calls setLCDD, but sets the RS line to low to send a command to the LCD.

`void clearLCD(void)`

Sends a command to the LCD to erase the display

`void clearPosLCD(uint8_t line, uint8_t pos, uint8_t length)`

Erases a specific part of the display. The parameters are: Line, start position in the line and length of the part to be erased.

Example:

```
clearPosLCD(0,10,5); // erases the last five characters
                     // in the first line of the display!
```

`void setCursorPosLCD(uint8_t line, uint8_t pos)`

Places the text cursor on a specific position on the display. The parameter line can be 0 for the top or 1 for the bottom line. The pos parameter must be between 0 and 15 for 2x16 LCDs.

`void writeCharLCD(uint8_t ch)`

Sends a single character to the LCD – this is identical to the writeChar function for the serial interface. However please make sure beforehand that the cursor is at the right position because otherwise you will not see the text!

Example:

```
setCursorPosLCD(1,5);  // position cursor in the second line, character 5
writeCharLCD('R');     // “RP6” is displayed, starting at
writeCharLCD('P');     // the cursor position!
writeCharLCD('6');
```

```
void writeStringLCD(char *string)
```

Like the relevant function for the serial interface, writeStringLCD sends a character string terminated by zero from the SRAM to the LCD. So you should use this function only if the text is dynamic and really located in the SRAM and is not plain static text. In that case, the macro:

```
writeStringLCD_P(STRING)
```

is more convenient as the text is read directly from the flash memory without the detour through the RAM (would consume a lot of memory).

```
void writeStringLengthLCD(char *string, uint8_t length, uint8_t offset)
```

This function allows to display a text of the specified length on the LCD. The parameters are identical to those of the same function of the serial interface.

```
showScreenLCD(LINE1,LINE2)
```

To make text outputs on an LCD easier, this function allows to write both lines of the LCD with only one query. The cursor is placed automatically at the right spot and the contents of the display are erased beforehand.

Example:

```
showScreenLCD("LCD line 1", "LCD line 2");
```

```
void writeIntegerLCD(int16_t number, uint8_t base)
```

This function is already known from the serial interface to display numbers in the formats BIN, OCT, DEC or HEX on the LCD.

```
void writeIntegerLengthLCD(int16_t number, uint8_t base, uint8_t length)
```

Except for the name, writeIntegerLengthLCD is identical to the well-known function on the serial interface. See the documentation in the RP6 manual and the comments in the code of the library for details.

4.1.5. SPI Bus

The SPI functions should NOT be used if a microSD card is inserted. They have just been ported from the RP6-M32 library and are not really necessary. Basically, you can connect further SPI peripherals to the ISP connector. You just need to use a chipselect pin from one of the other connectors and take into consideration that the active level shifters of the SD card are connected on the bus (via series resistors)! However it is better to use the UART / SPI ports for other SPI peripherals. The relevant functions to do that might get included in the library later on. Up to now, you can only refer to the AVR documentation.

```
void writeSPI(uint8_t data)
```

Transfers a data byte via the SPI bus.

```
writeWordSPI(uint16_t data)
```

Transfers two data bytes in a 16 bit variable via the SPI bus whereas the high byte is sent first.

```
void writeBufferSPI(uint8_t *buffer, uint8_t length)
```

Transfers up to 255 bytes via the SPI bus from an array of sufficient size. The number of bytes to be transferred from the array "buffer" are specified by the "length" parameter.

```
uint8_t readSPI(void)
```

Reads a data byte from the SPI bus.

```
uint16_t readWordSPI(void)
```

Reads two bytes from the SPI bus and sends them back as a 16 bit variable. The first read byte is the high byte.

```
void readBufferSPI(uint8_t *buffer, uint8_t length)
```

Reads up to 255 bytes from the SPI bus into an array of sufficient size.

4.1.6. ADCs

All ADC channels can be read via the function:

```
uint16_t readADC(uint8_t channel)
```

already known from the RP6Lib. There is not (yet) an automatic variant for the RP6-M256 that reads all ADC channels in sequence but you can add it yourself if you need it. Check the normal RP6Library for an example.

The channels are named differently as in the RP6Lib – 16 channels are available:
`ADC_15, ADC_14, … ADC_2, ADC_1 and ADC_0`

4.1.7. I/O Ports

As the RP6 CONTROL M256 features in total 60 free I/O ports, we will describe here how I/O ports of an AVR can be accessed in general. We will NOT describe hardware modules like timer capture, output modulator and others in detail. Please refer to the data sheet on how to use these modules.

The ATMEGA2560 has 9 I/O ports of 8 bit each plus one with 6 bit. Every port is controlled via 3 registers. One register is for the "direction" of the 8 I/O pins (DDRx) i.e. input or output, one register for writing (PORTx) and one register for reading (PINx).

If you want to use an I/O pin as an output e.g. to switch a LED, the relevant bit in the DDRx register must be set to 1, 'x' being the name of the port (A, B, C, … up to L (there is no PORTI)).

```
Example:
DDRL |= IO_PL5_OC5C; // PL5 is now an output
DDRL = IO_PL5_OC5C | IO_PL4_OC5B | IO_PL0_ICP4; // PL5, PC4, PC0 are now out-
puts, all other pins of PortL are inputs!


DDRE |= IO_PE2_XCK0_AIN0 | IO_PE4_OC3B_I4; // PE2, PE4 are now outputs
```

Via the PORTx register, the output can be set to high or low level.

```
Example:
PORTL |= IO_PL5_OC5C;  // High
PORTL &= ~IO_PL5_OC5C; // Low
```

If a bit in the DDRx register is set to 0, the relevant pin is an input.

```
Example:
DDRD &= ~IO_PD7_T2;  // PD7 is now an input
```

PINx register shows the status of the pin, i.e. if a high or a low level is applied to the pin.

```
if(PINC & IO_PC6)
      writeString_P("PC6 is HIGH!\n");
else
      writeString_P("PC6 is LOW!\n");
```

The built-in pullup resistors can be activated if the bits in PORTx register are set (only applicable if the port is configured as an input).

The ADC channels can also be used as I/O pins. Please note the different spellings compared to the names above (ADC_7 vs. ADC7)!

```
IO_ADC7, IO_ADC6,...
```

The definition of all I/O pins is in the header file RP6M256.h of the RP6M256Lib. Basically it is possible to use only numbers but names are often easier to identify.

Instead of e.g. IO_PE6_T3_I6 you can also write (1 << PINE6). This is identical and only defined like this in header RP6M256.h of the library.

By the way, after a reset, the bootloader configures all I/O ports on the expansion connectors as INPUT with pullup resistor. The only exception are the ports of the LCD connector. Those become only inputs if the display output of the bootloader has been deactivated. Please note: These are defined as outputs in the normal library! You might have to change this!

CAUTION:

The individual I/O pins are designed for a maximum current of 20mA. In total an 8 bit port should not be loaded with more than 100mA. The total load for all I/O ports must not exceed 400mA. If you want to switch bigger loads you have to do this via external transistors. All higher power electrical devices such as LED search-lights and motors definitely need an external driver circuit. They can NOT be connected directly to the I/O ports (a power driver and position control is already built-in model servo motors).

For more information please refer to the data sheet of the MEGA-2560 that is on the RP6 Web page / CD-ROM.

Prior to handling the module, please discharge yourself at a suitable grounded object (observe ESD safety precautions!). Please avoid short-circuits of I/O pins against the power rails or against each other. Reversed polarity can also damage the module.

4.1.8. Internal EEPROM

You can use the 4096 byte EEPROM that is integrated in the microcontroller, via two small helper functions. The EEPROM allows to store information in a permanent manner, the information will be kept even after power off.

With

```
uint8_t readINTEE(uint8_t adr)
```

a byte can be read from a specific address. Writing a byte to a specific address can be done with:

```
void writeINTEE(uint8_t adr, uint8_t data)
```

The address range includes 4096 bytes i.e. 0 to 4095 are possibles addresses.

CAUTION:

The first 32 bytes are reserved for the settings of the bootloader! So please do not overwrite the first 32 bytes (addresses 0 to 31) by mistake because the settings would be lost. This area is secured per checksum i.e. the bootloader recognizes wrong data and overwrites it with standard values!

Additional hint:

The EEPROM has a typical life time of several million write cycles. This is specified more accurately in the data sheet. Therefore the EEPROM should only be used to store settings in order to keep the number of cycles as low as possible (so better don't store a variable in the EEPROM 10x per second). It's preferable to store large data quantities on the SD card.

Tip:

The processor features some more hardware modules that haven't got any specific functions of their own in the library.

This could be added a later stage but will hardly happen for some quite specific modules (modulator, comparator,...). As usual you have to refer to the relevant chapter in the data sheet on your own and write your own control functions. Generally this makes more sense anyway because you can match it perfectly to your application.

4.1.9. MicroSD card

The microSD card can be controlled via an additional library that can be found in the RP6M256Lib directory. This library offers many functions to access partition and FAT16/32 file systems.

The library contains detailed documentation in HTML format in the directory:

RP6Lib/RP6control_M256_WIFI/sdc/doc/html/

The usage is briefly demonstrated in the example programs. Example program 4 shows how general information about the microSD card can be retrieved and how a file can be read.

In example 14, sensor data gets logged once every second into a file. The web server code in example 13 allows to retrieve the contents of the log file via a web browser.

CAUTON:

The microSD card slot is NOT intended for changing cards during operation! It might work but we can't guarantee for anything!

In general there is no warranty for any data losses! It is highly recommended to use an empty card.

Tip:

It might happen that the file system / partition table on the card gets damaged when a program error occurs or power off/reset happens while data is written to the card. It might be necessary to reformat the card e.g. with the Linux tool mkfs.vfat. In Windows Explorer, right click on the (correct) data storage device and select "Format...". In the worst case it might be necessary to set up a new partition table on the card e.g. under Linux via fdisk or under Windows via the disk manager. Of course all data on the card gets lost.

It is advisable to add a function to your own program which stops the writing process safely e.g. if one of the buttons is pressed, a specific command is sent or the battery voltage falls below a certain threshold. You should then always trigger this prior to turning the robot off.

This is a normal restriction for data storage devices with file a system. A PC could also damage the file system if you pull out the card during operation. Alternatively you can work in raw data mode on the AVR, this means without any file system at all. However then you can't use the card easily with your PC anymore...

4.1.10. WIFI Library

The WIFI Library offers a few helper functions to communicate with the WLAN module.

4.1.10.1. Data Communication

Basically sending data works almost exactly as with a normal UART, except that the functions have a _WIFI extension at the end:

```
void writeChar_WIFI(char ch)
void writeString_WIFI(char *string)
void writeStringLength_WIFI(char *data, uint8_t length, uint8_t offset)
void writeInteger_WIFI(int16_t number, uint8_t base)
void writeIntegerLength_WIFI(int16_t number, uint8_t base, uint8_t length)
```

To write and

```
void clearReceptionBuffer_WIFI(void)
uint16_t getBufferLength_WIFI(void)
char readChar_WIFI(void)
uint16_t readChars_WIFI(char *buf, uint16_t numberOfChars)
```

to read.

However there is a small specialty in the writeChar_WIFI function (and therefore in all other functions that use this function): It supports data flow control and might therefore block if the WLAN module takes more time to send a packet via the network. You must take this into account in your own programs.

For the rest, it works just as the functions already known from the RP6. Therefore we will explain only the additional functions here.

A small example should be sufficient:

```
writeString_P_WIFI("Hello Echo!\n");
```

Sends the constant text “Hello Echo!” with line break to the WLAN module that transmits it via WLAN. So the text will appear in the terminal of RobotLoader.

```
writeChar_WIFI('A');
```

Sends the ASCII character A.

```
uint8_t test = 42;
writeInteger_WIFI(test,DEC);
```

Sends the numerical value of the test variable in decimal format as ASCII text, in this case 42.

You will find many more details in the example programs. The reception of text is illustrated in the example programs and also explained in the other RP6 example programs.

Please also check the normal RP6 documentation and examples for further details on the UART functions which work almost identical!

4.1.10.2. Command Mode

The following functions are not necessary for the data communication with the PC. Once the module has been properly set up via RobotLoader, it becomes almost transparent for transmissions and can be used like a serial interface for the microcontroller. Some other advanced functions that the WLAN module offers, may be included in a future library version.

The command mode of the WLAN module is activated by the following function:

```
void enter_cmd_mode_WIFI(void)
```

Once this function has been executed, the WLAN module is in command mode. Depending on the firmware version running on the WLAN module, the change into command mode happens either by sending $$$ plus a 250ms delay, or alternatively (much faster) via an I/O port (GPIO14). Firmware version 2.32 and later introduce a GPIO cmd mode that can be activated in the source code of the library (configure the module accordingly, this function is normally deactivated). The newest version of the M256 library configures itself automatically via the configuration data that RobotLoader stores in EEPROM. Therefore you can simply activate the use of the GPIO14 pin in the WLAN configuration dialog in RobotLoader!

With

```
void leave_cmd_mode_WIFI(void)
```

you leave the command mode. You always have to leave the cmd mode before you can send data again.

Basically you can send commands to the module via the normal writeString / Char _WIFI functions – these must always be terminated by \r (=CR, Carriage Return). Be careful here, \n does not work! Alternatively you may also use \r\n.

In command mode, the module returns every received character as an echo. This can be used to check the correct transmission and is also useful to wait until the command has been completely received by the module.

If you want to send a command to the WLAN module and check the echo, you can use the function

```
int8_t writeCommand_WIFI(char * cmd)
```

However this checks only if the echo of the command has been received on return. The WLAN module sends every single character back immediately and this is checked by this function. This only makes sure that the command has been completely transmitted after having left the function. However a confirmation message or any other output is not checked.

The return value is 1 for successful and 0 for failure.

If you want to wait for an answer of the WLAN module, you can use one of the two following functions.

```
int8_t waitCharResponse_WIFI(char response, uint32_t timeout)
```

will wait for a single character (char response). And

```
int8_t waitResponse_WIFI(char * response, uint32_t timeout)
```

will wait for a complete string. For both functions you can set a timeout (is currently a simple counter without a fixed time reference).

You can also send commands by

```
int8_t issueCMD_WIFI(char * cmd, char * response)
```

to the WLAN module. The above function does a lot automatically. The command is sent and a reply is awaited. If this doesn't arrive, 3 further attempts are made. Return value is 1 if successful and 0 if failure.

All these functions are blocking, therefore you should use them with care! If a command fails (e.g. because the WLAN module was just busy with something else), long delays might occur.

For more complex programs, you might need non blocking variants. In this case you have to work manually with normal writeChar_WIFI and writeString_WIFI functions and integrate them into your own programs.

Tip:

You can switch the WLAN module into command mode via network connection or via serial interface. However only ONE of the two interfaces can be active at the same time, never both together.

The RobotLoader uses the network connection to send commands to the module (e.g. to trigger a reset of the microcontroller and query the signal strength).

You should build your own programs in such a way that you switch into command mode only for short periods and quit it as quickly as possible. Otherwise you might have to launch quite a few trials until the RobotLoader gains control over the module or you might even have to push the reset button manually.

In order to make the processing of text replies from the WLAN module easier, the following function allows to recognize and analyze individual lines:

uint8_t parseLine_WIFI(uint8_t data)

It must be called in a loop and every individual character must be passed to it. The return values of the function show if a complete line has been read. The line is stored in a buffer (= array) can can be processed further once it is complete.

The return value is:
0 if no complete text line has been recognized,
1 if a line delimiter has been recognized and
2 if the buffer was too small (everything that fitted in the buffer can still be read).
Standard is 254 characters.

The buffer is defined as

```
char receiveBuffer_WIFI[256]
```

and can be normally used in your own program.

The following simple example displays all text lines received from the WLAN module on the serial interface:

```
while(true)
{
	if(getBufferLength_WIFI()) // Received any data?
	{
		if(parseLine_WIFI(readChar_WIFI())) // Whole line?
		{
			writeString("\nReceived following text: ");
			writeString(receiveBuffer_WIFI);

		}
	}
}
```

If you now establish a connection to the module, you can enter text in the network terminal and send it. The text should then appear in the serial terminal. Of course you need to open the serial interface first.

Tip:

These few simple to use functions are already sufficient for working with the WLAN module.

Basically all special functions are driven by text commands and corresponding reactions of the WLAN module. The indicated functions allow to do that and therefore nothing else has been added to the central library up to now. The RP6M256Lib is focused on the basic communication, all the rest belongs to specific applications. Some functions are shown in the example programs but of course by far not all possible functions.

Small example: To change the IP address, netmask and gateway, just proceed as follows:

```
enter_cmd_mode_WIFI();
issueCMD_WIFI("set ip address 192.168.10.180","AOK");
issueCMD_WIFI("set ip netmask 255.255.255.0","AOK");
issueCMD_WIFI("set ip gateway 192.168.10.1","AOK");
leave_cmd_mode_WIFI();
```

You always have to wait for "AOK" which is the normal confirmation message of the WLAN module. Most of the other commands work the same way.

The basic configuration is already done by the RobotLoader and doesn't need to be done in your own programs. This saves time and space. Moreover you can change the settings without the need to recompile the program.

5. Example Programs

On the CD / website, you will find some very detailed example programs. They illustrate the basic functions of the RP6 CONTROL M256 WIFI. Just as for the robot, they may not always be the optimal solution. They are just a starting point for your own programs. This has been done on purpose such that there is some work left for you. It would be boring to just test ready-made programs...

The example programs are far from using all possibilities of the module! That means that there is much room left for your own extensions and much more complex programs. The biggest program is the selftest that uses just about 45kbyte memory (a large part of it is taken by the SDCard library). If you also consider the bootloader size, there is a rest of around 200kbyte memory which is enough for highly complex programs and even for small real-time operating systems and simple web servers. Moreover you can store very large amounts of data on the SD card.

You can of course exchange your own programs with other users via the internet. The RP6M256Lib and all example programs are licensed under the open source license "GPL" (General Public License) and therefore it is allowed to modify and publish the programs under the conditions of the GPL.

In general there is a multitude of example programs for the ATMEGA microcontroller family in the internet as these controllers are very popular with hobbyist users. However you must always match other example programs to the hardware of the RP6 CONTROL M256 and the RP6M256Lib – otherwise you might often encounter problems (the most obvious problems are different pin assignments, use of hardware modules that are already used elsewhere such as timers, different clock frequency, etc....) !

The following chapter briefly explains the functions of the example programs.

Example 1: "Hello World"-Program with LED text output and LED running light
Directory: <RP6Examples>\RP6_M256_WIFI_EXAMPLES\Example_01_LCD\
File: RP6M256_LCD.c

The program generates outputs on the serial interface

AND sends outputs via the WLAN connection!

So you should connect the robot to a PC and watch the outputs on the terminal of RobotLoader! As an option, you can connect the LCD!

The robot does *NOT* move in this example – *as long as you have loaded only the I2C bus slave program into the controller on the main board.* So you can put it on a table next to the computer.

This example displays a short "Hello World" text via the serial interface and via the WLAN module / the open network connection. You can see the output on both terminals.

Subsequently, a running light is executed. In addition, the LCD (if connected) will display first a static text and later a moving "Hello World" text – the two words "HELLO" and "WORLD" move slowly from one side to the other. A short break takes place after 16 seconds and this is displayed via the network connection. It resumes 8 seconds later.

Example 2: Buttons and LC Display
Directory: <RP6Examples>\RP6_M256_WIFI_EXAMPLES\Example_02_Buttons\
File: RP6M256_Buttons.c

The program generates outputs on the serial interface, the WLAN connection and on the LCD!

The robot does *NOT* move in this example program!

This example illustrates the use of the 2 buttons on the RP6 CONTROL M256. On every press of a button, the button number is shown on the display and via the serial interface and the WLAN connection.

Example 3: Analog to Digital Converter (ADC) and I/O ports
Directory: <RP6Examples>\RP6_M256_WIFI_EXAMPLES\Example_03_ADC_IO1\
File: RP6M256_IO_ADC.c

The program generates outputs on the serial interface, the WLAN connection and on the LCD!

The robot does NOT move in this example program!

Basically it works exactly as on the robot – but as you may need it very frequently for this expansion module, the example program shows how the free ADCs and I/Os can be used. The program shows also how simple it is to send telemetry data to a host computer.

Every 100ms a very long text message is sent per WLAN. The text length has been unnecessarily increased on purpose. This should show that it is also possible to send longer messages. The bandwidth can be used much more efficiently e.g. if a program on your PC should receive and display all sensor data permanently.

The text message transmits all I/O port statuses, all measurement values of the ADC channels and all stopwatch counter values. In addition to that a few 16 bit counter values in decimal and binary format (i.e. 16 ASCII characters are displayed – although 2 bytes would be sufficient – or alternatively 4 characters in hexadecimal format. As already said, not really efficient, just to increase the message size).

Example 4: microSD card
Directory: <RP6Examples>\RP6_M256_WIFI_EXAMPLES\Example_04_SDCARD\
File: RP6M256_04_SDCARD.c

The program generates outputs on the serial interface, the WLAN connection and on the LCD!

A suitable microSD card should be inserted into the card slot!

The robot does NOT move in this example program!

This program displays the contents of the test text file that should have already been stored in the main directory of the microSD card for the selftest program. It checks if a certain part of the text is read properly.

If it is not there already, please copy the file M256_SELFTEST_TESTFILE.txt from the example directory onto a suitable microSD card. You can then insert the microSD card in the card slot of the RP6-M256. The robot must be turned off when you do that! Then turn the robot on, upload and start the program. Some information about the SD card will be retrieved and then the contents of the text file will be displayed on the serial interface and per WLAN.

Example 5: WLAN commands
Directory: <RP6Examples>\RP6_M256_WIFI_EXAMPLES\Example_05_WLAN_CMD\
File: RP6M256_WLAN_CMD.c

The program generates outputs on the serial interface, the WLAN connection and on the LCD!

The robot does NOT move in this example program!

In some applications, it might be necessary to send commands to the WLAN module or to request some status information such as the current signal strength.

It is also possible to use various client applications for different protocols that are contained in the module firmware (HTTP Client, FTP Client...), but this is not described in detail in this manual. You will find further information on this subject in the documentation of the WLAN module.

This example program periodically triggers a scan for access points within range and displays the results on the serial and on the WLAN connection.

```
Example 6: I²C Bus Interface – Master mode
Directory: <RP6Examples>\RP6_M256_WIFI_EXAMPLES\Example_06_I2CMaster\
File: RP6M256_06_I2CMaster.c

This program shows the use of the I2C bus master mode. The controller on the
main board of the robot must have loaded the I2C slave example program (RP6-
Base_I2CSlave.hex)!
```

This program shows how to control the microcontroller on the main board in slave mode. This works only if the I²C bus slave example program from the RP6Base examples has been loaded into the controller on the main board.

Accessing the I²C bus works almost exactly as for the controller on the main board. The same functions are used.

All registers of the slave example are read and then transmitted per WLAN – and only per WLAN, not via the serial interface. A counter status is also displayed via the LEDs on the main board.

```
Example 7: I²C Bus Interface – Master mode – react to interrupts
Directory: <RP6Examples>\RP6_M256_WIFI_EXAMPLES\Example_07_I2CMaster\
File: RP6M256_07_I2CMaster.c

This program shows the use of the I2C bus master mode. The controller on the
main board of the robot must have loaded the I2C slave example program (RP6-
Base_I2CSlave.hex)!
```

Maybe you already noticed the interrupt signals on the XBUS connectors of the RP6? These can be used to react to sensor or status changes without permanently querying the slaves via the I²C bus as every bus access costs quite some time.

The ACS on the robot mainboard is a good example. The sensor status changes only quite *seldom* and it would not be very efficient to keep polling via the bus if something has changed. As soon as the status of the ACS changes, the slave example program will set the INT1 signal to high level. As INT1 is connected to one of the inputs of the MEGA2560 on the RP6 CONTROL M256, the controller can react immediately to this event and request the detail status of the controller on the main board via the bus.

However we don't use any interrupt routines to react to the event. Instead we poll the status of the pin once in every cycle of the main loop. As the I²C bus transfers are also interrupt controlled, a new transmission inside the interrupt routine is not possible. The task_I2CTWI() function that controls the process of the I²C transfer, must always be called in the main loop before any transfer happens. Therefore there is no point in using an interrupt routine. In fact, it would create problems because already running I²C bus transmissions could be interrupted. So, only the function task_checkINT() is used to poll the interrupt signal and eventually start a query. As soon as the status register 0 of the slave is read, the interrupt signal will be reset. The first three registers of the slave allow to find out what exactly has caused the interrupt.

This is exactly what the example program is doing. The outcome is that the current status of the ACS is shown via the 4 LEDs, the LCD and the WLAN connection.

At the beginning of the program, the transmit power of the ACS is set via the I²C bus. In addition to the ACS, the program also reacts on the bumpers and on eventual RC5 transmissions from a IR remote control or from other robots.

All other sensors of the robot can be handled in a similar way.

Another detail of the program is the "heartbeat" display. The task_LCDHeartbeat() function ensures that the character "*" flashes permanently at a frequency of 1Hz on the LCD. This is useful to determine if the whole program crashed completely or if only a certain part of the software is faulty. Once you write your own programs and it seems to crash completely, this function might be useful to identify the error source. A program crash can easily happen during the development.

Example 8: I²C Bus Interface – RP6 Master Library
Directory: <RP6Examples>\RP6_M256_WIFI_EXAMPLES\Example_08_I2CMaster\
File: RP6M256_08_I2CMaster.c

This program shows the use of the I2C bus master mode. The controller on the main board of the robot must have loaded the I2C slave example program (RP6-Base_I2CSlave.hex)!

As programs quickly become rather large and maybe even confusing for beginners, the ready to use RP6 master library will be used now for I²C bus communication. This has been reused mostly from the RP6-M32 module whose example programs show how it works in more detail and goes through it step by step. This library has been designed such that it can be used almost exactly as the normal RP6Lib for the controller on the main board. Many functions and variables have the same name as in the RP6Lib. This makes it easier to use parts of the programs for the RP6Lib with the RP6M256Lib. Even the event handlers for ACS, IRCOM and bumper are available again.

The I²C slave example program for the controller on the main board also contains a software watchdog function that works similar to the heartbeat function but via the I²C bus. If the master doesn't react within a determined time to an interrupt event (by simply reading register 0), all systems in the base unit of the robot are switched off and the program stops. And most importantly, the motors are turned off! Why do we have to do this? Imagine the master controller sends the command to drive forwards at 10cm/s right before the software crashes, then the robot will continue to move on even if collides with an obstacle. The watchdog stops the robot in such cases. The software watchdog timer is normally deactivated. A command must be sent via the I²C bus to activate the watchdog. It is also possible to configure the watchdog in such a way that it triggers an interrupt event every 500ms in order to check if the master controller is still alive. We will use this function in the next example.

There are also event handlers for watchdog requests and low battery status.

The rest of the program is similar to example 7. The only new feature is the additional watchdog timer whose requests will also appear on the LCD. Moreover all sensor registers are read and displayed via the serial interface. As before, ACS, bumper and RC5 events are displayed.

Example 9: I²C Bus Interface – Motion functions
Directory: <RP6Examples>\RP6_M256_WIFI_EXAMPLES\Example_09_Move\
File: RP6M256_09_Move.c

This program shows the use of the I2C bus master mode. The controller on the main board of the robot must have loaded the I2C slave example program (RP6-Base_I2CSlave.hex)!

CAUTION: The robot moves in this example program!

Now we will test some of the move functions already known from the RP6Lib, that are now remotely controlled via the I²C bus: move, rotate, moveAtSpeed, changeDirection and stop. The functions can be used the same way as those in the RP6Lib. In this example we removed everything from the previous examples except for the watchdog display. This was required in order to keep it simple and to use the blocking mode of the move functions (a few things like the heartbeat display wouldn't work anyway with this). In this example, the robot moves forwards and backwards and rotates at about 180° - exactly like in example 7 of the RP6Lib ("RP6Base_Move_02.c").

Example 10: I²C Bus Interface – Behavior based robot
Directory: <RP6Examples>\RP6_M256_WIFI_EXAMPLES\Example_10_Move\
Files: RP6M256_10_Move.c

This program shows the use of the I2C bus master mode. The controller on the main board of the robot must have loaded the I2C slave example program (RP6-Base_I2CSlave.hex)!

CAUTION: The robot moves in this example program!

With the new library, the behaviour example programs from the RP6 base can almost be reused 1:1 without many changes. This is exactly what has been done with the example program "RP6Base_05_Move_05". Only a few small changes were necessary. Among others, the LEDs on the main board must be controlled via function setRP6LEDs as setLEDs is already dedicated to the LEDs on the RP6-M256...

Otherwise the program is almost identical to the already known program – the robot drives around and avoids obstacles. The only difference is that this time it is controlled from the RP6-M256 module.

Showing the currently active behavior on the LCD and via the LEDs is new. With this additon you can immediately see which behavior is currently active. A small helper function ensures that the text is only sent once to the LCD. Otherwise the text would flicker on the display. While the "Cruise" behavior is active, the 4 red status LEDs execute a simple running light.

The battery status is also monitored. If the battery charge level gets very low, the robot stops. However this takes quite a while with a freshly recharged battery...

Example 11: WLAN remote control 1
Directory: <RP6Examples>\RP6_M256_WIFI_EXAMPLES\Example_11_WIFI_REMOTE_1\
File: RP6M256_11_WIFI_REMOTE_1.c

This program shows the use of the I2C bus master mode. The controller on the main board of the robot must have loaded the I2C slave example program (RP6-Base_I2CSlave.hex)!

Moreover the robot is remote-controlled via the WLAN connection (text commands)

CAUTION: The robot moves in this example program!

The WLAN connection is perfectly suitable for controlling the robot remotely. How this can be implemented is shown in this program.

It is a very simple command interpreter that can react to specific text commands. However the command interpreter is directly built into the behavior based model from the other examples so that the program can be easily extended with autonomous behaviors at a later stage.

In this example behaviour_wifiControl is the only active behavior.

A little instruction manual of the program is displayed right at the start per WLAN. First you type in "cmd" in order to activate the interpreter. "Activate" means activation of the robot control or triggering the activation of the behavior. The interpreter is always waiting for commands. If it is "active", the robot can be controlled via the interpreter.

The keys w-a-s-d and q, e are assigned to the direction control. For every command you have to press the Enter button, for example:

```
60 + Enter  … the robot drives forwards

d + Enter … the robot turns right, the command interpreter automatically
reduces the speed during turns because otherwise it would be extremely
quick compared to the forward movement.

s + Enter … Backwards.

120 + Enter … faster

w + Enter … forward again

q + Enter … left turn

Enter … Stop!   (alternative 0 + Enter)

z + Enter … quit Interpreter
```

While the interpreter is running a lot of sensor values are transmitted via WLAN. In this case every 100ms. This could be used for a graphical display on the remote control PC. The text commands could also be bound to a graphical user interface with buttons or joystick / gamepad control.

Example 12: WLAN remote control 2
Directory: <RP6Examples>\RP6_M256_WIFI_EXAMPLES\Example_12_WIFI_REMOTE_2\
File: RP6M256_12_WIFI_REMOTE_2.c

This program shows the use of the I2C bus master mode. The controller on the main board of the robot must have loaded the I2C slave example program (RP6-Base_I2CSlave.hex)!

Moreover the robot is remote-controlled via the WLAN connection (text commands)

CAUTION: The robot moves in this example program!

The autonomous behavior of the other programs is added again here and you can activate and deactivate the behaviors via WLAN or override the behaviors and control the robot manually.

The Escape behavior is always active because if the robot collides with an obstacle, it is always better to drive back and around the obstacle rather than continuing. However the command mode has a higher priority. This means that we can for example let the robot push a box on purpose without activating the escape behavior.

Avoid and Cruise can be activated and deactivated. Cruise should be activated as last step because once this is activated the robot just drives forward without checking for obstacles. Even if Cruise is deactivated, the robot still reacts on obstacles (just try to move your hand in front of the robot or press on the bumpers).

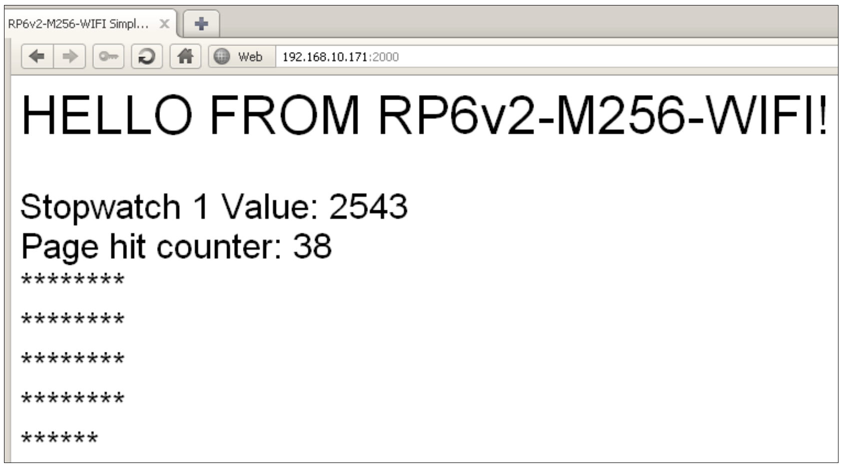
As soon as any behavior except the “stop” behavior is active, all sensor values of the robot are transmitted at 10Hz via WLAN.

Example 13: Very simple web server
Directory: <RP6Examples>\RP6_M256_WIFI_EXAMPLES\Example_13_Simple_Webserver\
File: RP6M256_13_Simple_Webserver.c

The program generates outputs on the serial interface, the WLAN connection and the LCD!

In this example program the robot *does not* move!

The network data connection can also be used for applications like web servers. This example illustrates this in an very simple way. It implements an extremely minimalistic web server. The code for the server is just about 25 lines short (without HTML contents and debug outputs).



HELLO FROM RP6v2-M256-WIFI!

Stopwatch 1 Value: 2543
Page hit counter: 38
****
****
****
****
******

The web server is only able to do one single thing which is sending the web page on the screen shot to the querying browser.

The page is dynamic though, i.e. the program may change the contents. It simply displays the value of a stopwatch (will be reset to 0 after 10 seconds). Upon every page request a counter is incremented and a corresponding number of * are displayed. Only one connection at a time is possible. Therefore the page shouldn't be updated too quickly nor should 2 browsers access it simultaneously. The web server always reacts with the same reply to the GET query. It is unable to deliver other files such as pictures. It would be possible but would need some more effort. Moreover with bigger files such as pictures, the limited transmission speed must be taken into account.

Basically, the example can also be extended with forms for input possibilities which would allow to control the robot per web browser.

Example 14: Data logging to SD card + web server
Directory: <RP6Examples>\RP6_M256_WIFI_EXAMPLES\Example_14_SDCARD_logging\
File: RP6M256_14_SDCARD_logging.c

The program generates outputs on the serial interface, the WLAN connection and the LCD!

In this example program the robot does not move!

The SD card will now be used to store data. 1x per second all sensor values are queried from the robot and their values are written into a text file on the card. Among others, a few standard C functions for the processing of the strings will be used.

Additionally, the small web server from the previous example has been integrated in the program to enable retrieving the stored data via a web browser.

Please observe the previously mentioned safety precautions for the microSD card! Every writing access must be terminated by pressing the SW1 button (running light stops and a message is displayed on the LCD) before the robot is turned off or a reset (also via WLAN connection) is triggered.

Another option to handle that has been integrated in the web page: it can either start or stop the logging via two links. The query of the browser is checked for the page requests start_log or stop_log. The web page is displayed normally and dynamically whether the logging runs or not.

We have come to the end of this additional manual. The provided example programs are only a basis / starting point. Now you can give free rein to your own creativity, write new programs and install new sensors on the RP6 that you can control via the RP6-M256. Or do something completely different with it. The module is quite universal and can also be used to control other devices, not only robots.

APPENDIX

A – Pin Assignments

This section contains the pin assignments of the main connectors with the I/O ports.

The connector of the serial interface has exactly the same pin assignment as on the main board. Of course this also applies to the two XBUS connectors.

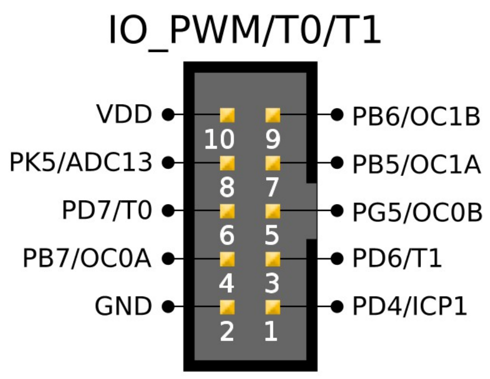
I/O Ports:

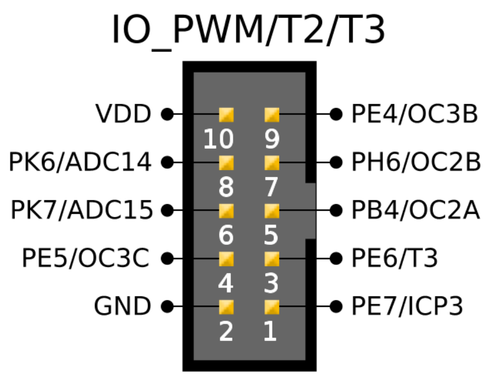
All 10-pin I/O port connectors (and the 10-pin ADC port, see next page) are mostly pin compatible. This means the 5V power supply and the I/Os are on the same pins but the additional functions may be on different pins. As far as possible the alternate functions have been assigned in a similar way e.g. the two SPI / UART ports are on the same pins. Most of the PWM and timer/capture channels are also on the same pins, however the timers themselves are different (note: OC0B and OC2A/B are 8 bit timers, the others are 16 bit. OC0A has a dual assignment with OC1C).

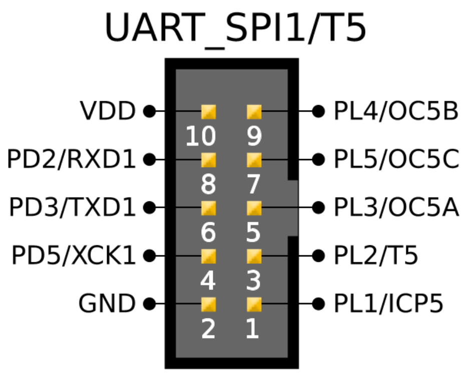
This allows to set up several pin compatible sensor and actuator PCBs.

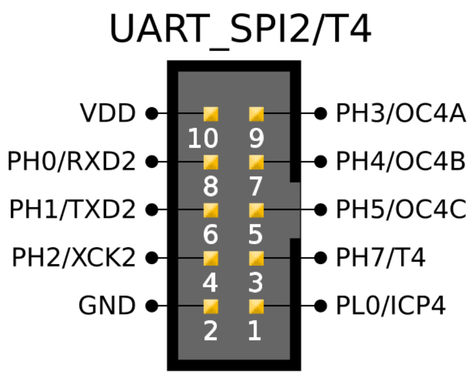
CAUTION: Avoid to connect the power supply pins of another expansion module (e.g. an experimentation module) to the power pins of these connectors. Use the XBUS instead for power supply.

The power supply pins are not designed for direct operation of high power consumption devices! Small servo motors are acceptable with additional filter capacitors (>=100µF). Several bigger servos must be powered directly from the RP6 main board, via a separate battery or via a separate 5V voltage regulator.





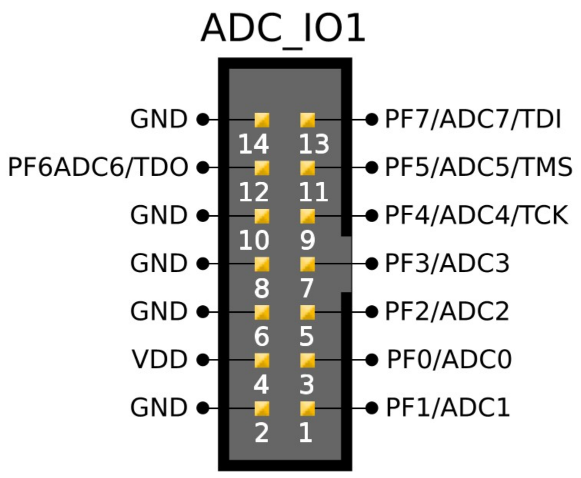


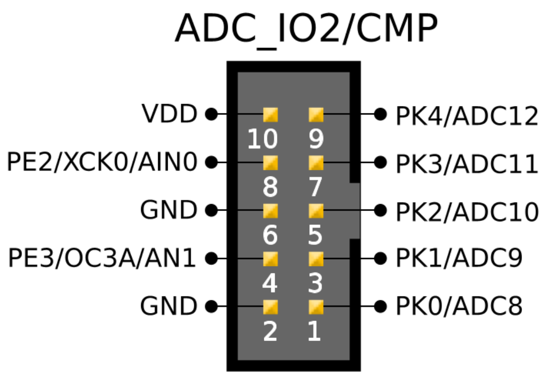


ADC Channels:

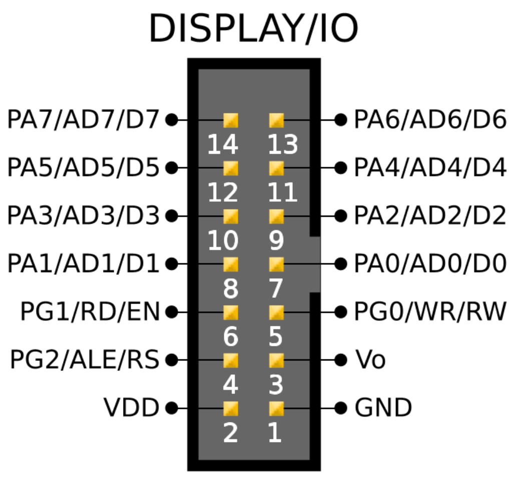
Out of the 16 ADC channels (that can also be used as I/O pins) 8 are available on an 14-pin and 5 on a 10-pin connector, along with the 5 V operating voltage. The remaining 3 channels are spread on the other 10-pin connectors (see previous page).

The 10-pin connector offers two additional normal I/O ports with analog comparator and OC3A PWM channel alternate functions. The 14-pin connector also contains the JTAG interface (apart from the reset signal, see ISP connector).





LCD Connector:



If you don't want to use the standard LCD, this pin assignment helps you building your own cable for the LCD.

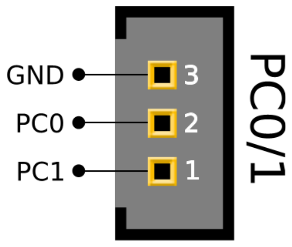
Unlike the RP6-M32 and RP6-M128 module, the display can be controlled in 8 bit mode and no external SPI shift register is used for the data bits.

Thus the pins can also be used as normal I/O ports. A part of the XMEM memory interface is brought out here that might eventually be used with extra I/Os for graphic displays and other specific applications (not tested).

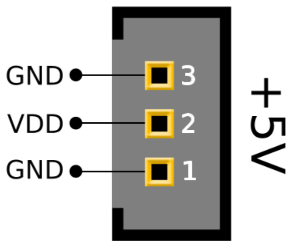
Make sure that the pin assignment is correct and that the plug is not connected mirrored!

The designators of the individual pins can vary from one manufacturer to the other (we are talking about the part after the last / of the pin designators. The designators before are the port names of the ATMEGA), but usually the designators are identical to those used here and you can connect the pins 1:1.

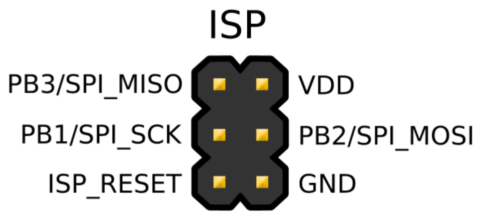
Other Connectors:



PC0/A8 and PC1/A9 are available on a small 3-pin connector. The remaining address pins of the XMEM interface are used for other purposes.



On another 3-pin connector you will find the 5V supply rail to connect more external devices e.g. for displays with different back-light (CAUTION: might require a series resistor or current source). If the RP6-M256 module is used without the RP6, you may connect the whole module to an external 5V power supply here without having to use the XBUS connector.

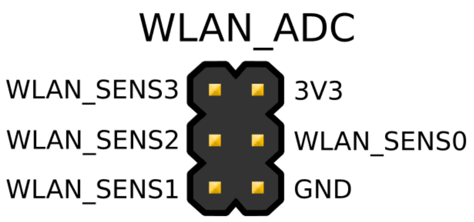


The 6-pin ISP connector (In System Programming) is meant for advanced users.

Beginners: DO NOT USE!

(at least not without a good reason)

It is connected in parallel to the microSD card slot. Therefore the I/O ports (and thus also the ISP adapter) should only be used if no SD card is inserted. The pins contain active 3.3V level shifters (via series resistors s. circuit diagram). You should take this into account when you connect your on circuits to them (which pins are input/output).



4 A/D converter channels of the WLAN module are available on another 6-pin connector just next to the WLAN module. They allow usage for different applications and can be read by the WLAN module independently from the microcontroller. The WLAN module can be woken up from standby and the measurement values can be requested via network in command mode. The A/D channels are connected via a resistive voltage divider in order to extend the measurement range (normal: 1.2V max, 0.400mV measurement range, 500mV wakeup trigger ; voltage divider factor: 0.12821 ; at the connector: 9V max, 0..3.2V measurement range, 3.9V wakeup trigger).

The connector also brings out the 3.3V power rail. This connector is also intended for rather advanced users. For normal applications you should use the A/D channels of the microcontroller.

The two jumpers are marked clearly on the PCB and thus they are not shown here. The ISP/BOOT jumper allows to switch the reset pin between the ISP connector and the normal RP6 reset signal. The ADHOC jumper can activate ADHOC mode and reset the WLAN module to the default settings if it is bridged prior to powering on the unit (there is no jumper bridge included in the package as it only has to be bridged for a very short time → you can use e.g. a screwdriver or the ISP/BOOT jumper).

If necessary an external reference voltage for the A/D converter can be connected to the two pin AREF connector (the AREF pin is identified on the PCB, the other one is GND).

B - Tips for the WLAN Module

1. WiFi Module Passthrough Mode

As explained in chapter 3.4, the passthrough mode can be activated via the menu item

```
Options → WiFi SERIAL Config → Enable Passthrough Mode
```

To do this, the USB interface must be connected and the correct port selected in the serial flash loader. After that, everything that is entered into the serial terminal is transmitted to the WLAN module. Check in the options dialog that CR or CR+LF is activated! The WLAN module needs at least one CR after every command, LF alone doesn't work.

You can also use the menu item

```
Options → WiFi SERIAL Config → Reset WIFI + Enable Passthrough
```

This triggers a hardware reset of the WLAN module and the complete boot output is displayed. The output in the serial terminal should look approximately like this if the module has been correctly configured and can connect immediately to the access point:

```
WIFI:
WiFly Ver 2.32, 02-13-2012 on RN-171
MAC Addr=00:06:66:71:e6:05
Auto-Assoc M256TST chan=11 mode=WPA2 SCAN OK
Joining M256TST now..
*READY*
Associated!
Using Static IP
Listen on 2000
```

If anything has been wrongly configured or the access point is not in range, the output will rather look like this:

```
WIFI:
WiFly Ver 2.32, 02-13-2012 on RN-171
MAC Addr=00:06:66:71:e6:05
Auto-Assoc M256TST chan=8 mode=NONE FAILED
*READY*
Auto-Assoc M256TST chan=8 mode=NONE FAILED
Auto-Assoc M256TST chan=8 mode=NONE FAILED
```

Type in:

```
.$$$
```

to switch into command mode.

It makes sense to first scan for access points with the command

```
scan
```

It lists all access points within range and displays some of the configuration parameters. If your own access point is not in the list, it might be too far away or the reception is disturbed by something (check the antenna!). The reason might also be a faulty configuration of the access point. Execute the command SEVERAL TIMES as it might happen that the access point is not listed immediately.

2. Firmware Update and Restore Factory Settings

The firmware on the WLAN Module can be updated via an FTP server from the manufacturer where the module downloads the new software automatically. The old firmware remains in the Flash memory and can easily be re-activated in case of problems. A firmware update is even possible with the RobotLoader as it contains a small FTP server for the test routine that is activated during a very short period during the test. This function should become generally usable in future versions.

Until this function is implemented, the firmware must be properly configured (gateway!) and an internet connection must exist to update the firmware. Start by checking which version is installed in command mode:

```
ver
```

With

```
ls
```

the firmware images stored in the flash will be displayed.

```
FL# SIZ FLAGS
 11  19   3 WiFly_EZX-2.23
 30   1  10 config
 51  20   3 WiFly_EZX-2.36
185 Free, Boot=11, Backup=51
```

If you received e.g. a module with firmware 2.23, you can update it to the current version 2.36 that contains new useful functions (e.g. activate the command mode via GPIO – that is quicker).

Key in

```
ftp update wifly7-236.img
```

or just simply `ftp update`, and the most recent image is downloaded. The module loads the firmware and activates the boot image.

If this has been executed successfully, type in

```
reboot
```

to reboot.

After the reboot, it is recommended to restore the default settings and re-do all settings completely – that can be done very conveniently with the RobotLoader GUI.

First of all, click on

```
Run Factory RESET!
```

in the tab Configure Scripts. Wait until the process is finished. Then click on "Run initial WIFI config (BAUDRATE IS 9600)!" and wait. Watch the outputs in the log tab. A green "OK!" should appear as a reply to several of the commands.

The normal baud rate of 9600 is increased to 500000 bauds.

If this worked, you can repeat the configuration in the General Settings tab – don't forget to set the WPA pass-phrase!

If the new version causes problems, you can reactivate an old image with

```
boot image XX
```

XX stands for the number that the "ls" command displays in column #FL in front of the file name. So, according to the example above

```
boot image 11
```

for the old 2.23 firmware and

```
boot image 51
```

for the newer one 2.36.

Then reboot and run a Factory RESET.

If you get an error like "530 Login Authentication failed" or similar when trying to update the firmware, it may be that your module uses older factory default settings. The FTP address for updates has changed recently and must be manually set correctly with the following commands:

```
set ftp address 0
set dns name rn.microchip.com
save
```

If this still does not work, check for more recent infos which you can find online.

3. Usage of GPIO for Command Mode

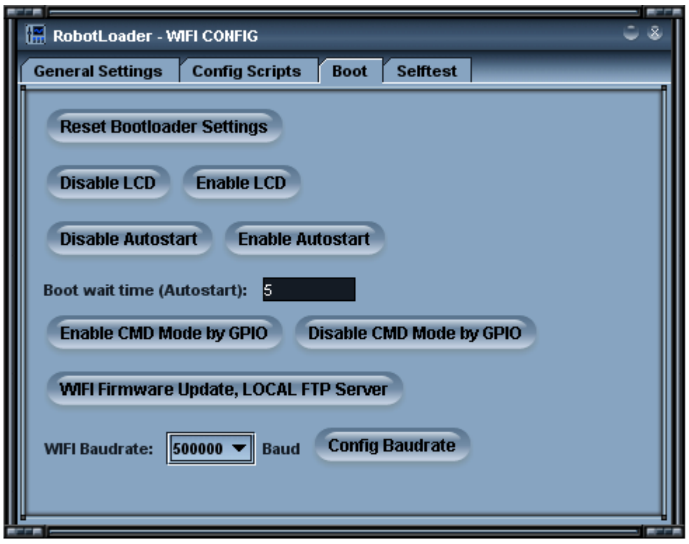
Firmware Version 2.32 or newer allow to use a GPIO to switch into command mode. This is much faster than waiting for 250ms, sending $$$, waiting again 250ms and checking if the command mode has been activated.

To activate this, just execute

```
set uart cmd_GPIO 14
save
reboot
```

(case sensitive!). GPIO14 can be controlled via the microcontroller.

In RobotLoader versions newer than 2.3b, this can also be done in the WLAN configuration dialog in the "Boot" tab. Click there on the button "Enable CMD Mode by GPIO" and it will be activated.



Boot wait time (Autostart): 5

WIFI Baudrate: 500000 Baud

The RP6-M256 library automatically analyzes the setting in the EEPROM of the microcontroller and decides which mode is active. So nothing needs to be changed in the library.

In the same dialog, you can also enable automatic start of the program (after an adjustable delay) as soon as the unit is connected to a power supply, and deactivate the LCD. Additionally the baudrate of the WLAN module (and the microcontroller UART that is connected to it) can be configured. This setting is stored in the WLAN module and in EEPROM of the microcontroller and the user programs will automatically use that baudrate.

WARNING:

DO NOT change the baudrate without any good reason! You may have to do a manual factory reset using the ADHOC Jumper if something goes wrong here...

4. ADHOC Mode and Reset of the Module to Factory Settings

The module contains a jumper between the UART_SPI1/T5 connector and the antenna that is labeled "ADHOC".

If this jumper is bridged DURING (=before and during) powering the robot on, the ADHOC mode is activated and most of the settings are reset to factory defaults. You can find more detailed information in the documentation of the manufacturer. You can use this mode to configure the module without the serial connection. However this is not supported by the RobotLoader, you have to work directly with the command line interface. To connect to the module via ADHOC mode is not as simple as with normal infrastructure mode – the setup procedure is highly dependent on the installed WLAN software and operating system. Therefore this is not described any further in this section.

If the jumper is bridged immediately after power on and you toggle it 1x per second 5x in sequence (not faster than 1x per second), the module will be totally reset to factory settings. You should see that the LEDs change the blink rate once you bridged it often enough.

After that, the initial configuration can be done in RobotLoader.

5. List of Tested Routers / Access Points

The following routers / access points / series have been tested with the WLAN module and confirmed to be compatible so far (end of 2012). The list is not exhaustive. If a router is not listed here, it doesn't mean that it wouldn't work but that no test we know of has been made so far.

- Linksys/Cisco WRT series
- Cisco Aeronet
- LANCOM L-322agn
- T-Com Speedport W701V with original firmware
- T-Com Speedport W701V with AVM Fritzbox 7170 firmware
  (Hardware manufacturer is AVM)
- AVM Fritzbox WLAN 3270
- Netgear WGR614 v8
- Netgear WGN54
- DLINK DIR-615
- DLINK DIR-600
- Airlink 101
- Apple Airport express
- Buffalo Networks

C – Solving Configuration Problems

WLAN networks offer many different configuration options. Therefore it might happen that some settings create problems with the configuration of the WLAN module.

The following list contains a series of proposals to solve problems with the WLAN connection of the RP6v2-M256-WIFI module – not everything will be applicable to your specific problem. If a problem occurs, please work completely through the list, check if everything is correctly set and watch carefully how the WLAN module behaves during the execution of the various instructions. If the problem persists, please contact the customer support and provide all relevant infos!

1. Do you use the RobotLoader version 2.3c / 20120717 (or later)? This is important for the correct configuration of the module.
2. Do you use static or dynamic IPs (DHCP)? If you use DHCP, please try to change to static IPs (caution: either switch off or change the configuration of other computers in the network. Alternatively connect ONLY ONE test computer linked directly per cable to the router).
3. Are the router, PC and WLAN module all in the same IP subnet? E.g. all computers in subnet 192.168.10.x with subnet mask 255.255.255.0
4. Does the WLAN and network connection work with other devices? If the WLAN connection doesn't work with a notebook or similar devices, it will most likely also not work properly with the WLAN module.
5. All other network settings are as shown in the instruction manual? Especially the following items should be checked:
   - Safety: WPA2 PSK correctly configured in the router (avoid WPA1/WPA2 mixed mode, some routers don't transmit the safety settings properly).
   - Is the pass-phrase really identical in the router and the WLAN module? Problems may arise with special character $ and spaces as these are control characters of the command interpreter in the WLAN module. Normally spaces can be replaced by $ and the RobotLoader does this automatically in the latest versions. If $ and spaces occur in the pass-phrase at the same time, you have to use another replacement character. This can be configured. However the most simple way is not to use $ and spaces in the pass-phrase... all other special chars should work.
   - Please operate the WLAN network in the 2.4GHz band, 5GHz is not supported!
   - Data exchange between WLAN devices must be allowed.
   - Switch off Network repeater / WDS functions.
   - Deactivate MAC filter.
   - Switch OFF DHCP.
   - Check all other functions and eventually also the advanced options.

6. With AVM routers, the option:

   `“The listed WLAN devices are allowed to communicate among each other”`

   must be activated. If this option has been changed, it may take some time until the connection works. If necessary, turn the router and the WLAN module off and on again (the router might need up to two minutes for the reboot).

7. If settings are changed, some router models need some time until the changes become effective or may even require a restart. However after a restart, it should work. The first connection may take a bit longer. The following connections should be very quick though.

8. Is a firewall active on the router or the computer (also check the Windows firewall), that might block the data exchange between the WLAN module and a computer? Please check carefully the settings of all involved firewalls!

9. Port 2000, TCP and UDP must be allowed! In addition port 55555 UDP should be allowed (for Broadcast and Auto Discover function).

10. Please try and deactivate the firewall completely.

11. Check all settings of the WLAN router! Is anything blocked somewhere? Also check the logs of the router to see if something blocks while the WLAN module is connected.

12. Please open the serial(!) terminal in RobotLoader and click on
    `Options->WIFI Serial Config->Reset WIFI + Enable Passthrough`
    in the RobotLoader menu. The module must of course be connected via cable. Then type in the following (always confirm with Enter)

    ```
    .$$$
    ```
    (Dot and 3x Dollar signs)

    Now the command mode should be active. This will be confirmed with “CMD”.

    ```
    show rssi

    show net

    show status

    get wlan

    get ip

    get comm

    get uart

    scan

    scan

    scan
    ```
    (3x scan with a bit longer pause in between to make sure that all networks within the range will be listed)

    ```
    ping 192.168.10.x
    ```
    (type in the IP of an active computer in the network)

    ```
    exit
    ```

Copy the whole output and send it per mail to the customer support! In case of problems with marking text in the terminal: double-click inside the terminal, keep pressed at the second click, then drag, STRG+C, STRG+V...)

Alternatively send the complete log file of RobotLoader (see below)

13. Open the command line of the operating system (Windows: `Start->Execute` then enter `cmd`)

    and execute

    ```
    ping 192.168.10.x
    ```

    with the IP of the WLAN module. Do the pings get received/acknowledged?
14. Is the WLAN module displayed in the `Options->Discover WIFI Devices` dialog in RobotLoader (after power on this might take up to 20 seconds)?
15. Unplug the USB interface from the module and try to connect again. Also check the jumper ISP/BOOT (should be set to BOOT location).
16. Restart the router/access point (turn it off and on again, wait for 3 minutes and then switch the WLAN module on).
17. Check if the WLAN connection works with other devices (Notebook, Smartphone or similar)
18. Try with a notebook (or equivalent) to connect to the WLAN module. The notebook should be connected PER WLAN to the same access point, not per LAN cable. If this works, only the data exchange from WLAN to LAN is blocked.
19. Is the access point within range or are there too strong sources of interference nearby? Try to change to another WLAN channel (of course in the 2.4GHz band).
20. If WPA1 or WPA2 is used, it might happen that the router needs some more time to establish the connection. The WLAN module is fitted with a timer that counts down and by default interrupts the connection attempt after one second You can lengthen this to 5 seconds as follows:

    ```
    .$$$
    set opt jointmr 5000

    save

    reboot
    ```

    Then try again. If it doesn't solve the problem, try to increase it to 10000.
21. If WPA1/WPA2 MIXED mode is active, please try to use only WPA2 or alternatively WPA1. Some routers transmit the active mode in an unexpected or incorrect way. If both don't work, try again with WEP128 or OPEN. WARNING: WEP and OPEN are unsafe and therefore only meant for a short test, just to see if it works in general!

22. If nothing helped so far, THEN (<u>and only then</u>) try a factory reset of the WLAN module. Go into the configuration dialog in tab "Config Scripts" and click in sequence on the buttons:

    1. `Run Factory RESET!`
    2. `Run initial WIFI config (BAUDRATE is 9600)!`
    3. `Run default WIFI config!`

    Wait every time until the scripts are completely finished! Then redo the WLAN settings in the "General Settings" tab and try again to connect.

23. This does not reset 100% of all settings. If there are still problems or if the serial interface has been set to an unknown baud rate by mistake, the module can also be reset 100% to factory settings if you proceed as follows:

    1. Turn the power supply off.
    2. Touch a metal part of a grounded device (connected to protective earth building connection, water tap, PC with metal chassis) to discharge yourself because of the risk of electrostatic discharge (ESD).
    3. Bridge the ADHOC jumper on the WLAN module (small screwdriver) and keep it bridged!
    4. Switch on the power supply. The LEDs of the WLAN module should now flash faster than during the normal boot process!
    5. Immediately afterwards bridge and open the contact with a screwdriver or a jumper (e.g. the jumper of the ISP/BOOT) 5x in sequence with pauses of one second (not shorter).
    6. The LEDs should flash slowlier once this has worked (and they also do so after a timeout so this is not a sure sign if it really worked...)
    7. If it doesn't work, try again (and this time with one and a half second pauses)!
    8. Continue with 22. step 2..

Other solutions that are rather a fallback solution if nothing else works:

24. Try with another PC or notebook
25. Use the ADHOC connection – e.g. with a USB WLAN stick or notebook
26. Execute a factory reset of the access point / router (PRIOR to that export all settings into a file if possible and get your internet provider access data in order to be able to set them correctly even without the backup file, also write down the other most important router settings). Then perform ONLY the network and WLAN settings and NOTHING else. Then try it again with the WLAN module. Save settings / export to file. Now make all other settings and retry. If it worked before you made all settings, then you may compare (diff) the two exported files and check which settings could block the connection...
27. Try another router / access point if available.

General Information about Support Requests:

If problems occur that are not described in the manual or if the proposed solutions don't work, please contact the customer support with an accurate and detailed description of what has been done and how it has been done. Due to the complexity of the product, it is difficult to provide good assistance without detailed description of the problem.

On every new start, RobotLoader generates a new log file in the program directory which is extremely helpful for most of the problems.

If you tried several things in RobotLoader and reproduced relevant error messages, terminate RobotLoader and send the full log file:

```
<RobotLoader>/logfiles/robotloader_logfile.log
```

as attachment to your request to the customer support.

Warning: The file is overwritten on every new start of RobotLoader and all previous information gets lost. So it is recommended to copy the file into another folder right after terminating the program!

Please don't mix up the log file with the "Log" tab in the RobotLoader window! This is NOT the log file. The log tab only shows some important status messages. The log file contains a lot of very important extra information.

Moreover it could also be helpful if you report how the LEDs behave on the robot in case of hardware problems: which LEDs light up / flash directly after power on and how do they behave if you click on "Reset", or "Hold in Reset" or "Connect" in the RobotLoader (serial loader tab)?

D – Recycling and Safety Instructions

Recycling

Do not throw the RP6 or any expansion modules into normal household waste! Like all electrical devices, these components must be brought to a recycling depot or any other collection point for used electrical devices!

If you have any questions, please contact your dealer.

Safety Instructions for Batteries

All types of batteries must be kept out of the reach of children! Do not leave batteries within the reach of children as children or pets might swallow them. Please immediately seek medical assistance if a battery has been swallowed!

Leaking or damaged batteries can cause burns if they come in contact with skin. Use in this case appropriate protection gloves to handle the battery. Do NOT short-circuit batteries and do not throw them into fire. Do not recharge normal batteries! They might explode! Recharge only rechargeable accumulators e.g. NiMH batteries with an appropriate charger.

Recycling Instructions for Batteries

Do not throw accumulators or batteries into normal household waste! You as an end consumer are legally obliged to bring back all used batteries and accumulators to your dealer or a recycling center. It is prohibited to throw them into household waste!

Bring faulty or empty accumulators and batteries back to your dealer or to a collection point for batteries. You can bring used accumulator and batteries to any battery sales point.

By following these rules, you comply with your legal obligations and you contribute to the protection of the environment!