PocketBot

a matchbox-sized line following robot

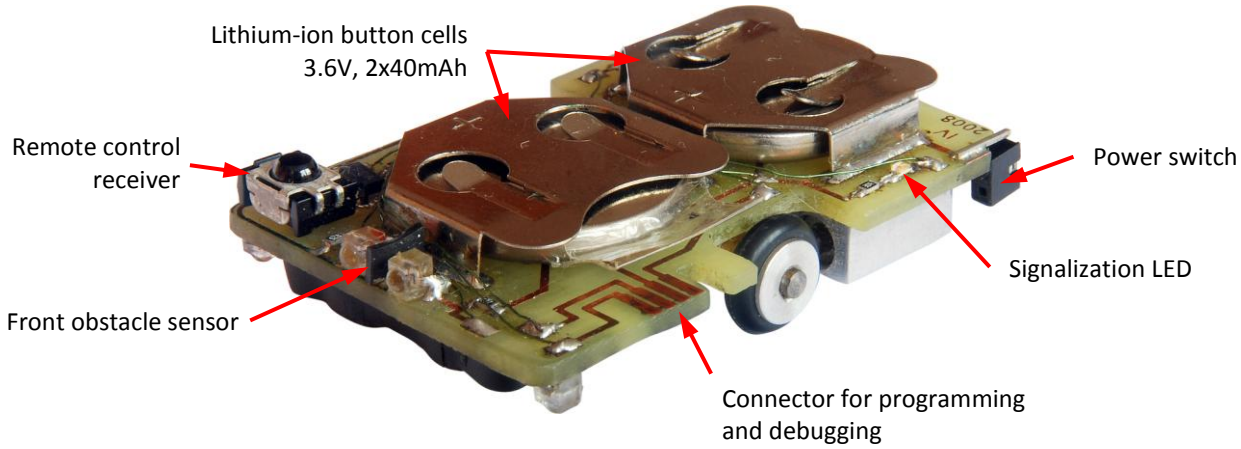
PocketBot project consists of three parts. The key part of the project is the robot itself – a tiny line following vehicle of a matchbox size. Furthermore, the robot is supported with an USB communication device and with a PC control application. Altogether, these three parts form a complex solution to the line following issue. Each part of the project will be described in this article.

PocketBot – the line following robot

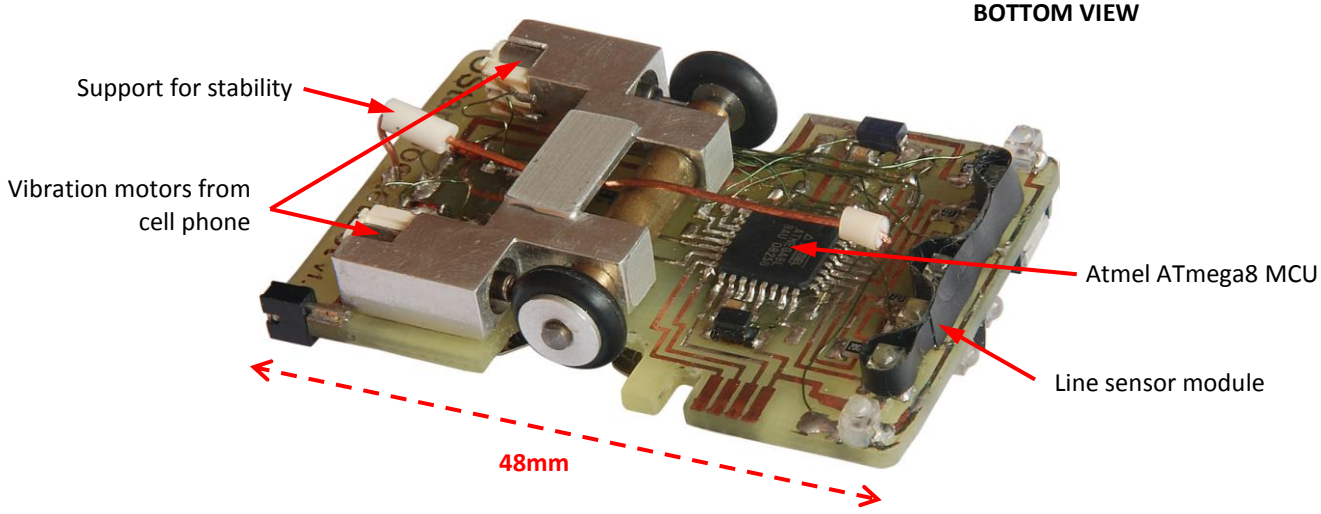
The robot was primary designed to fit into a matchbox. A homemade double-sided printed circuit board stands as the robot's chassis at the same time. Robot is powered with two rechargeable lithium-ion button batteries wired in parallel (3.6V, 40mAh each). The Atmel ATmega8 microcontroller runs robot's program, which is written in C. An 8-pin connector offers ISP and UART interface for programming and debugging, respectively.

dimensions:	***48 × 32 × 12 mm***
weight:	***19 g (body 13g, cells 6g)***
speed:	***0.35 ms^{-1} (line following)***
	0.6 ms^{-1} (maximal)

TOP VIEW

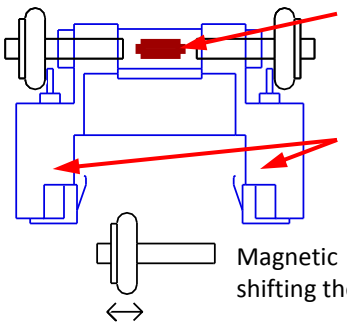


BOTTOM VIEW



Undercarriage

Two separately driven wheels (8mm diameter) provide differential steering. The dimensions of the gear mechanism were crucial due to considerable space constraints. Fortunately, I met Josef Vandělík who designed and manufactured the wheelframe for my robot. [3] The wheelframe employs a friction gear system with magnetic pressure. A neodymium magnet in the central tube attracts wheel axles, pressing each wheel to the motor shaft.



Neodymium magnet

Two vibration motors from cell phone propel the robot

Magnetic pressure can be adjusted by shifting the wheel on its axle.

Line following

The robot is capable of line following. That means it is able to follow a black guiding line marked on bright surface. There might be line crossing on the track, in such case the robot chooses a straight direction. The robot can avoid obstacles on the way; if an obstacle is detected, the robot reverses and continues backwards.

Finally, the robot can find a guiding line in unknown environment. When there is no line present under the robot, the robot starts to seek on a spiral trajectory until it crosses a guiding line.

Line sensor module

A guideline is marked with a black PVC isolation tape. This material doesn't reflect infrared light so it is easy to distinguish a guideline by a light reflection of the surface.

The sensor module consists of 3 detectors and 4 emitters. The emitters (infra-red LEDs) and detectors (phototransistors) are placed in a row alternately so that each phototransistor is surrounded with two IR LEDs. Thanks to this design it is possible to measure the surface reflexivity on six spots under the sensor module, using only three phototransistors and four IR LEDs. Generally, this approach reduces the number of components and ADC inputs required for a line sensor module, which is desired with respect to dimension constraints.

The illustration bellow shows how this method works:

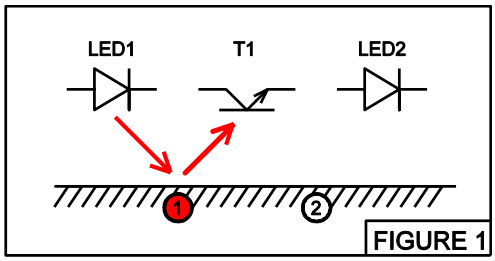


FIGURE 1

LED1 is emitting infrared light that reflects to the phototransistor T1, hence the light reflexivity at point 1 is measured.

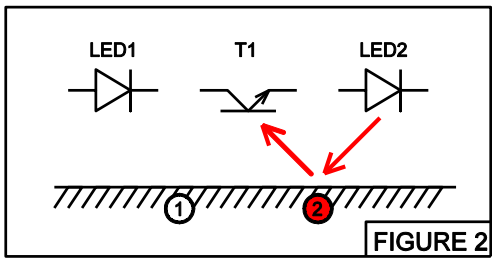


FIGURE 2

Then, LED1 is turned off and LED2 starts emitting IR light. The phototransistor T1 measures the light reflexivity at point 2.

In real application (especially when high refresh rate is desired) the characteristics of IR components must be taken into account. Due to reaction delays (latency), data from sensor might be biased if the reflexivity at point 2 is measured right away after measuring the reflexivity at point 1. To keep from this, I analysed sensor characteristics on an oscilloscope and then I modified the scanning sequence in a way so that the sensors do not interact with each other.

Ambient light suppression

Because light conditions often vary according to time and place, it is necessary to use an ambient light suppression algorithm for sensors to work properly. The method is simple: Every sensor does two measurements. At first, it scans for the amount of ambient light. Then, it turns its infra-red LED on and measures the value again. Subtracting these two values, the bias of ambient light is suppressed.

Sensor calibration

There might be slight differences in characteristics of individual optical components; therefore the sensor module should be calibrated. The calibration is done manually in two steps:

1. Offset calibration

All sensors are placed above the black guiding line. Once the calibration command is received, all sensors measure the surface reflexivity and measured values are stored in memory. Those are the offset calibration values. From now on, all measured values are automatically corrected with this offset. (Each time a measurement is made, the offset is simply subtracted from the actual value). As a result, all sensors will return equal value when they are located above the black line.

2. Gain calibration

During the gain calibration all sensors are placed above a white surface. Some sensors might be more sensitive than others, so the measured values differ from each other. But because the surface reflexivity under the sensor module is supposed to be equal, the gain coefficients for each sensor can be easily calculated. For future measurements, every measured value will be corrected (multiplicated) with its gain coefficient; so that all calibrated sensors will have similar characteristics.

Consequently, the calibrated sensor module will output normalized values.

Processing data from sensors, motor control

Optical sensors measure the light reflexivity of the surface and acquired data is processed by the line detection algorithm. The algorithm is designed in such a manner that line width doesn't matter. The line detection algorithm outputs signed integer value that states the actual deflection of a guideline. Values close to zero mean that the line is located accurately in the middle of the sensor module, positive values state how much does the line deflects to the right and negative values state the deflection to the left.

This output is then used for proportional-integral-derivate (PID) control of the line tracking. The PID controller adjusts the motors' speed according to the actual line deflection and previous states. The position of the line is evaluated 30 times per second.

In other words, the PID controller drives the robot so that the line is always centered to the middle of the sensor module, so that the robot performs smooth line following.

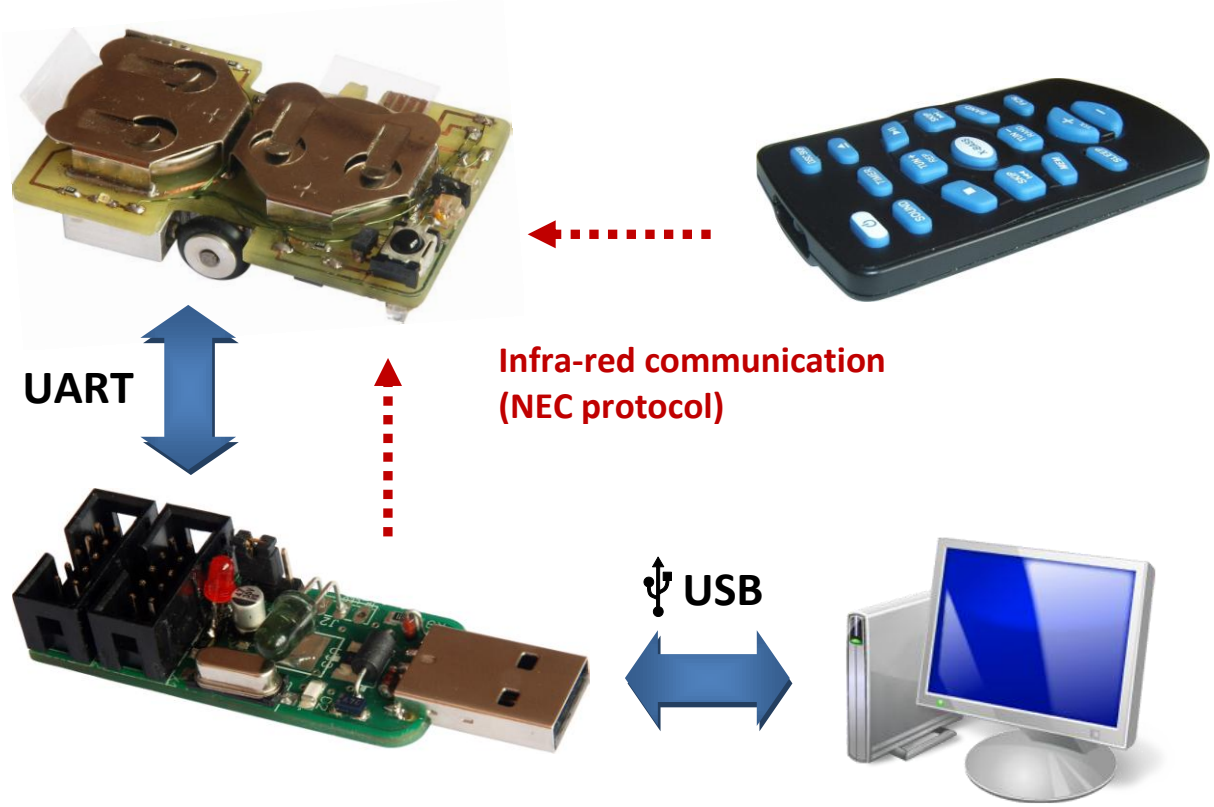
Remote controlling

Robot is equipped with an infra-red remote control receiver. Therefore it can be controlled with a standard remote control or from a PC application. The wireless link to the robot is used for adjusting parameters (such as speed and PID constants), for sending sensor calibration commands and also for a manual operation. The wireless communication utilizes the NEC remote control protocol. [4] This protocol was implemented in both PocketBot (as a receiver) and USBdockStation device (as a transmitter).

USBdockStation

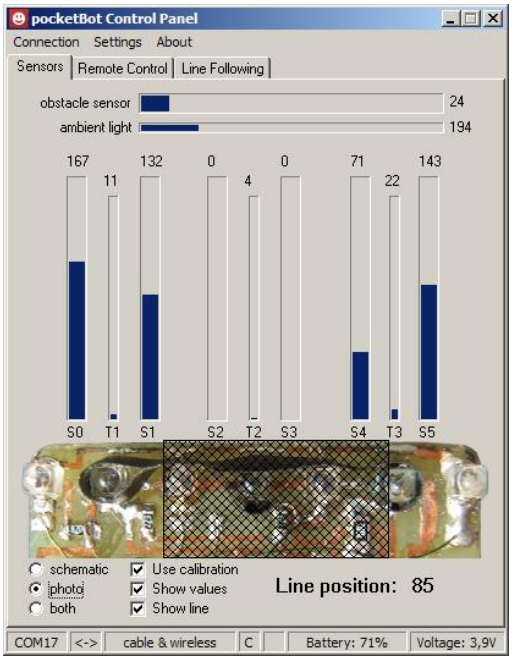
The USB device ensures both wire and wireless communication between the computer and PocketBot. It's based on an AVR-CDC project, a USB to UART converter. [2] Once the device is connected to a computer, operation system creates a virtual COM port which can be accessed from a computer application. PocketBot, USBdockStation, and the PC application communicate with each other through this UART interface, using a particular protocol that was designed for this purpose.
To control PocketBot remotely, I added some extra functionality to the original AVR-CDC firmware: The USBdockStation can emulate an infra-red remote control. It has a power infra-red LED and it is capable of sending IR NEC remote control packets as an ordinary remote control. It is obvious that the wireless communication is only unidirectional.
Firmware is programmed in C and it runs on ATmega8 with external 16MHz oscillator. The printed circuit board has custom design to fulfill needs of my project and it was manufactured industrially.

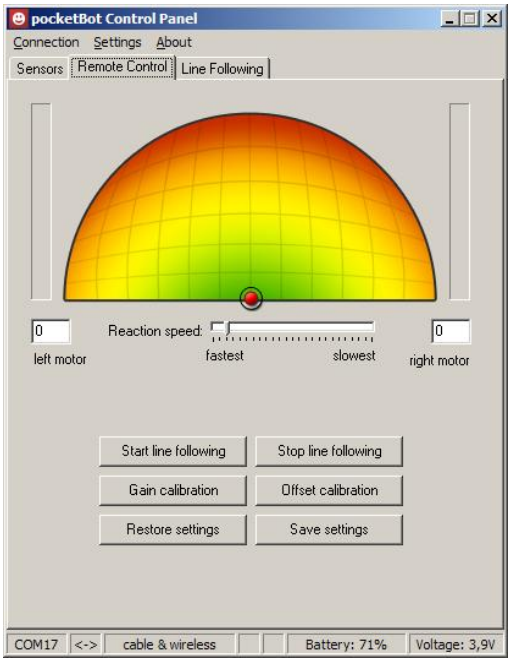
Communication diagram



PocketBot Control Panel

The PC application offers sensor diagnostic; it shows real-time visualization of sensor module state, including the assumed line position. Moreover, the application has also capabilities of sending wireless commands to the robot, which gives the key feature of adjusting PID controller constants remotely. Thus, PID constants can be tuned on-line during line following. On the other hand, wireless communication can be used for manual operation as well.
After startup, the application automatically scans over all COM ports for USBdockStation presence. Then it can detect whether the robot is connected with a cable, or whether only wireless communication mode is available. In bi-directional (cable) mode the battery fitness and calibration data can be accessed.
The application is programmed in Borland Delphi and runs on Windows operation systems.





Project timeline in brief

2007/06 I have started programming microcontrollers.
2007/10 I have designed and put up together my first line following robot [9]
2007/12 Project PocketBot started. I got inspired by the Desktop Line Following robot [1]. I built sensor module prototype and I designed the line position algorithm. Ambient light suppression method tested.
2008/01 Sony remote control protocol decoding, UART debugging, PWM motor control.
2008/03 Robot's undercarriage finished
2008/04 Lithium-ion charger built. I won first place at the Student Scientific Competition with my first line following robot.
2008/05 PocketBot linefollower finished. Line following with proporcional (P) regulator [5]
2008/06 I presented paper titled Line following robots [11] at the international conference VIASL [10]
2008/07 PD regulator allows line following at higher speed [6]
2009/02 I implemented new NEC remote control protocol into the robot.
2009/04 USBdockStation finished. Sensor gain and offset calibration. Working on optimized switching sequence for sensors.
2009/05 PocketBot Control Panel finished. Graduation at high school in Prague.
2010/01 Remote control protocol analyzer finished [7]
2010/03 PocketBot won first place in the freestyle category on RobotChallenge competition in Vienna. [8]

Author

Ondřej Staněk (20) is the first year student of the Computer science at the Faculty of Mathematics and Physics at Charles University in Prague, the Czech Republic. He is interested in computer programming and electronics. Apart from mobile robotics, the author participates in a development of a snow measuring device for hydrometeorology science.
www.ostan.cz, ostan89@gmail.com

References

[1] ChaN's Desktop line following robot: http://elm-chan.org/works/ltc/report.html
[2] Software USB to UART converter for AVR microcontrollers: http://www.recursion.jp/avrcdc/
[3] Author of pocketBot's wheelframe: http://www.volny.cz/jova3/pocket_bot/pocket_bot.htm
[4] NEC protocol description: http://www.sbprojects.com/knowledge/ir/nec.htm
[5] PocketBot video I.: http://vimeo.com/6394938
[6] PocketBot video II.: http://www.youtube.com/watch?v=8UCQyGJ5M0E
[7] Remote control protocol decoding application: http://ostan.cz/IR_protocol_analyzer/
[8] http://www.robotchallenge.org/scripts/trunk/trainstation/detail.php?hide_controls=true&refresh=99999&competition=126
[9] My first line following robot: http://ostan.cz/robot/ipage00013.htm
[10] The VIASL conference: http://www.ifip2008praha.cz/
[11] Submission paper for the VIASL conference: http://www.ostan.cz/robot/line_following_robots.doc