E-puck knihovna pro Python

David Marek

Univerzita Karlova v Praze

5. 4. 2011

Osnova

1. Představení e-puck robota
2. Připojení
3. Komunikace
4. Přehled příkazů
5. Ukázky programů
6. Závěr

Představení e-puck robota

E-Puck

- Ecole Polytechnique Fédérale de Lausanne
- Miniaturní robot (průměr 75mm)
- Opensource hardware
- Spousta senzorů
- Bluetooth komunikace
- Dostupný (v labu)

Senzory a akční členy

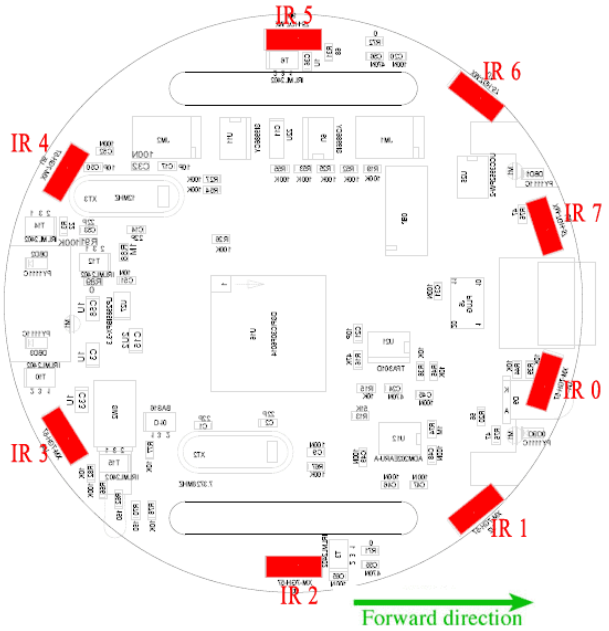
- Kamera (640x480)
- IR senzory (proximity / ambient light)
- Akcelerometr (3D)
- Mikrofony
- LED
- Krokové motory
- Speaker

Kamera

- Rozlišení 640x480
- Dva režimy: RGB565 / barvy šedi
- Příliš velké fotky pro zpracování v robotovi
- Řešení problémů s pamětí:
  - Prokládání
  - Změna formátu fotografie
- Reálně jde získat:
  - 40x40 barevně
  - 55x55 černobíle
  - Lineární kamera

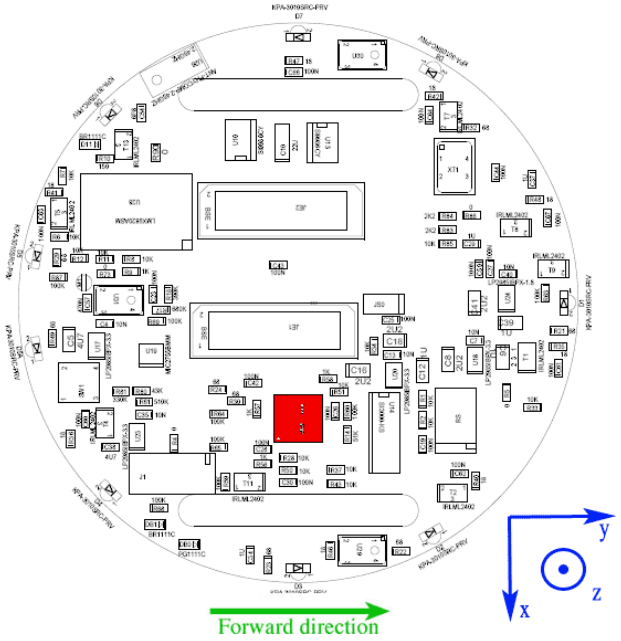
IR senzory

- 8 senzorů po obvodu
- Dva režimy:
  - Aktivní – rozeznávání překážek
  - Pasivní – intenzita světla
- Překážky do $\sim$ 4cm



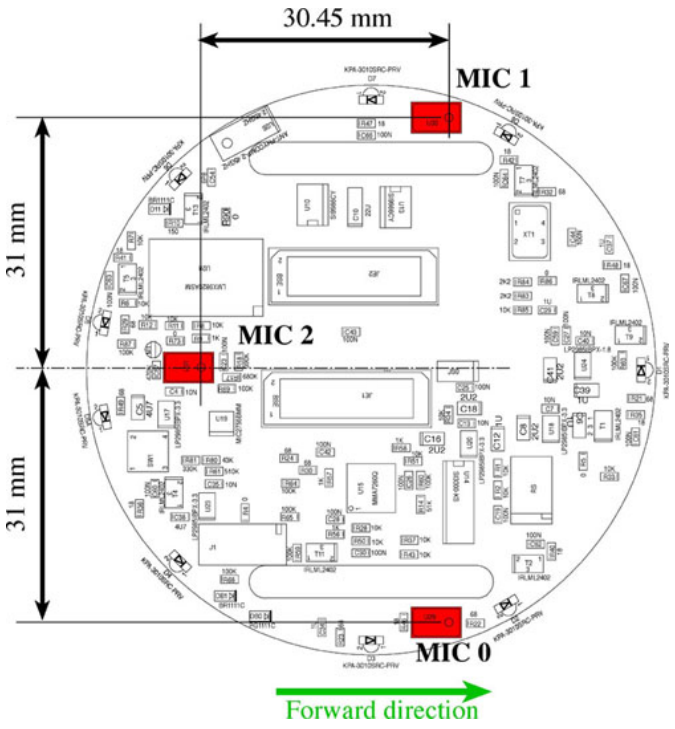
Akcelerometr

- 3D akcelerometr
- Z robota se dá získat:
  - Vektor akcelerace
  - Zpracované informace (rotace, zrychlení, . . .)
- Detekce nárazu, naklonění, směru pohybu



Mikrofony

- 3 mikrofony
- Vlevo, vpravo, vzadu
- Lze získat:
  - Hlasitost
  - Frekvenci (FFT)

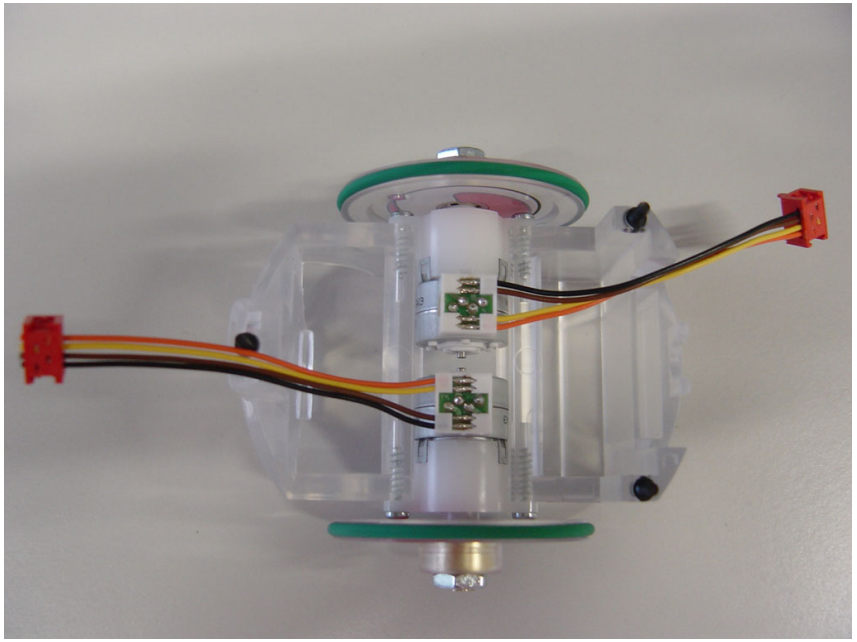


LED

- 8 LED po obvodu robota
- Zelená dioda v těle robota
- Jasná dioda u kamery

Krokové motory

- Dva krokové motory
- Tisíc kroků = jedna otáčka kola
- Jeden krok $\approx$ 0.13 mm
- Maximální rychlost jedna otáčka za sekundu
- Neklouzavá úprava



Speaker

- Přehrává připravené zvuky (WAV)
- 5 zvuků zakompilovaných do firmware

Připojení

Příprava PC

- Potřeba funkční bluetooth
- Balíky `bluez-firmware`, `bluez-utils`
- Utilita na zaslání PIN (`bluez-pin`, ...)

Nastavení rfcomm

Získat adresu robota

```
$ hcitool scan
Scanning ...
10:00:E8:52:C6:3E e-puck_1055
```

Nastavení /etc/bluetooth/rfcomm.conf

```
rfcomm2 {
    bind no;
    device 10:00:E8:52:C6:3E;
    channel 1;
    comment "e-puck_1055";
}
```

Vytvoření spojení

Připojení pomocí `rfcomm`

```
# rfcomm connect rfcomm2
Connected /dev/rfcomm2 to 10:00:E8:52:C6:3E on channel 1
Press CTRL-C for hangup
```

Možné chyby

Can't connect RFCOMM socket: Connection refused

- Není spuštěná aplikace, která by předala PIN

Can't create RFCOMM TTY: Address already in use

- Robot už je připojen
- Jiná aplikace s ním stále komunikuje

Nahrání firmware

- Používá se `epuckupload`
- Je potřeba mít aktivní připojení
- Robot dokáže přijímat nový firmware pouze pár sekund po restartu

Postup při nahrávání

1. Spuštění nahrávání

```
$ epuckupload -f BTcom.hex rfcomm2
```

2. Restartování robota
3. Vyčkání na konec nahrávání
4. Robot připraven k práci

Komunikace

Komunikace

- Kontrolní program běží v PC napsaný v Pythonu
- Příkazy jsou textové (binární) řetězce posílané přes sériové rozhraní
- V robotovi je upravený firmware BTcom, který pouze zpracovává příkazy
- Je potřeba aby bylo vytvořené připojení

Příklad ovládání

Příklad zaslání příkazu

```
>>> from epuck import Controller
>>> controller = Controller("/dev/rfcomm2")
>>> controller.set_speed(100, 100)
>>> controller.get_speed()
(100, 100)
```

1. Je vytvořen Controller, parametr je cesta k rozhraní kde je robot připojen
2. Nastavena rychlost obou motorů na 100
3. Zjištěna rychlost z robota

Jde o synchronní komunikaci, program byl vždy zablokován dokud nedošla odpověď od robota.

Druhy komunikace

Synchronní

- Vždy víme, zda se příkaz provedl
- Transparentní komunikace
- Hned se dozvíme výsledek
- Pokud nastala chyba tak může zůstat v deadlocku

Asynchronní

- Není třeba čekat, než se příkaz vykoná
- Spolehlivá komunikace
- Zotavování z chyb
- Musíme si zvlášť říct o výsledek

Asynchronní komunikace

Příklad programu

Ukázka kódu

```
from epuck import Controller

controller = Controller("/dev/rfcomm2", asynchronous=True)
photo_request = controller.get_photo()
while not photo_request.response_received():
    foo()
photo = photo_request.get_response()
```

- Místo odpovědi získáme jen handler
- Je možné zkontrolovat zda-li už přišla odpověď
- Než přijde odpověď, tak je možné dělat cokoli jiného

Asynchronní komunikace

Callback

- Pro zpracování odpovědi je možné mít speciální funkci
- Každý příkaz má parametr callback, což je funkce
- Callback je zavolán hned po přijetí odpovědi
- Callback může být použit pro modifikaci odpovědi:

```
vysledek_get_response = callback(odpoved_od_robota)
```

Ukázka callback metody

```
from epuck import Controller

controller = Controller("/dev/rfcomm2", asynchronous=True)
controller.get_photo(callback=lambda img: img.show())
```

Přehled příkazů

Příkazy

- Třída `Controller` obsahuje příkazy pro ovládání všech sensorů a akčních členů robota
- Každá akce je pouze zavolání jedné metody
- Při synchronní komunikaci vrací metody rovnou výsledek
- Při asynchronní komunikaci vrací handler.
- Bližší informace v dokumentaci

Příkazy

Přehled

- Motory
  - `set_speed(left, right)`
  - `get_speed()`
  - `set_motor_pos(left, right)`
  - `get_motor_pos()`
- IR senzory
  - `get_proximity_sensors()`
  - `get_ambient_sensors()`
  - `calibrate_sensors()`
- LED
  - `set_body_led(on)`
  - `set_front_led(on)`
  - `set_leds(on)`
  - `set_led(led_no, on)`

Příkazy

Přehled

- Kamera
  - `set_camera(mode, width, height, zoom)`
  - `get_camera()`
  - `get_photo()`
- Přepínač na robotovi
  - `get_turning_switch()`
- Akcelerometr
  - `get_accelerometer()`
  - `get_raw_accelerometer()`
- Zvuk
  - `play_sound(sound_no)`
  - `get_volume()`
- Restart nebo zastavení
  - `reset()`
  - `stop()`

Ukázky programů

Následování světla

- Využity IR senzory
- Robot se pohybuje za světlem
- Je potřeba zdroj IR světla (svíčka, . . .)

Zdrojový kód 1/3

```
import logging
import time
import sys
import signal

from epuck.controller import Controller

logging.basicConfig(level=logging.WARNING)

c = Controller("/dev/rfcomm2", asynchronous=True)
```

Následování světla

Zdrojový kód 2/3

```
# Signal handler to cleanup after shutdown
def handle_signal(signum, frame):
    r = c.stop()
    r.join()
    sys.exit(0)

# SIGINT is interrupt signal sent by CTRL+C
signal.signal(signal.SIGINT, handle_signal)

while True:
    # Read ambient light sensors
    r = c.get_ambient_sensors().get_response()

    front_side = r["L10"] + r["R10"]
    left_side = r["L45"] + r["L90"]
    right_side = r["R45"] + r["R90"]
```

Následování světla

Zdrojový kód 3/3

```
# The light source is straight ahead
if front_side < left_side and front_side < right_side:
    c.set_speed(500, 500)
elif left_side < right_side:
    c.set_speed(-500, 500)
else:
    c.set_speed(500, -500)

time.sleep(0.1)
```

LED

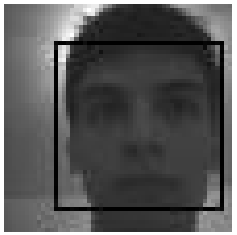
- Pouze efekty s LED
- Přepínač na robotovi využit pro různé režimy

Vyhýbání se překážkám

- Jednoduchý program na principu Braitenberg vehicle
- Výstup na akčních členech je ovlivněn pouze hodnotou senzorů

Detekce obličejů

- Použita kamera a OpenCV knihovna
- Program každých pár sekund sejme obrázek a snaží se na něm nalézt obličej



Evoluce

- Experiment s knihovnou PyEvolve

Závěr

Odkazy

Stránky e-puck

http://www.e-puck.org

Bootloader

http://svn.gna.org/viewcvs/e-puck/trunk/tool/bootloader/computer_side/multi_platform/

Dokumentace

http://e-puck.davidmarek.cz

Projekt

http://github.com/davidmarek/E-Puck-Algorithms-Library