Remote control for CPG based robots

Final Presentation - 18 June 2010

Gabriel Cuendet
3rd year Bachelor in Electrical Engineering

Assistant: Alessandro Crespi

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

EPFL Biorobotics Laboratory
Plan

• 20’ Presentation
• 10’ Questions
• 10’ Demo, using the robot
The goal of this project is to develop a wireless remote control that communicates with a robot, which is controlled by a CPG. The electronics for the RF part aren’t part of that project, since they are already designed. The remote control is able to configure a small number of locomotion parameters on the robot. It allows a user to interactively remote control the robot without needing a PC.
Goal of the project

Gamepad + PC + Radio interface

To the robot
Goal of the project

Gamepad + Radio interface + power supply

AUTONOMOUS DEVICE
Parts of the project
(state at mid-term presentation)

- **Hardware**
  - User interface
  - Battery management

- **Software**
  - User interface’s PIC18F2580

- **Housing and mechanical integration of the hardware**

  - Dev. Version
  - In progress…
Hardware (*reminder*)

- Battery protection, charger and 3.3V DC/DC converter
- Radio interface (developed by A. Crespi)
- User interface
User interface
Parts of the project

- **Hardware**
  - User interface
  - Battery management

- **Software**
  - User interface’s PIC18F2580

- Housing and mechanical integration of the hardware
Final ver. vs Development ver.

- Possibility of a supplementary button
- Removal of the on-off LED
- Reset of the radio interface by software

\[
U_{\text{max}} = 0.2 \cdot V_{DD} = V_{DD} \cdot \frac{R_{\text{max}}}{47 \cdot 10^3 + R_{\text{max}}}
\]

\[
R_{\text{max}} = \frac{0.2 \cdot 47 \cdot 10^3}{1 - 0.2} = 11750 \Omega
\]
Parts of the project

• Hardware
  • User interface
  • Battery management

• Software
  • User interface’s PIC18F2580

• Housing and mechanical integration of the hardware
Functions (functions.c / .h)

- Registers operations
  - Getters and setters of various length
Functions (functions.c / .h)

• Registers operations
• Display functions
  • Operations on the LCD (I²C)
Functions (functions.c / .h)

- Registers operations
- Display functions
- Battery management
  - Getters for the battery voltage, temperature and current (\(I^2C\))
Functions (functions.c / .h)

- Registers operations
- Display functions
- Battery management
- Robots locomotion
  - Start, stop and update functions for each kind of robot
Functions (functions.c / .h)

- Registers operations
- Display functions
- Battery management
- Robots locomotion
- User interface
  - `press_button(char button)`, `display_menus()`, ...
Functions (functions.c / .h)

- Registers operations
- Display functions
- Battery management
- Robots locomotion
- User interface
- Misc
  - sync(), err(), scan()
Functions (functions.c / .h)

- Registers operations
- Display functions
- Battery management
- Robots locomotion
- User interface
- Misc
- Menu functions
  - Functions to fill the menu table: void function(void)
Menu

- INFOS RC
  - Battery voltage
  - Radio interface version
  - Reset radio interface microcontroller (PIC16F)
  - Reset user interface microcontroller (PIC18F)
Remote control for CPG-based robots -
Gabriel Cuendet

Menu

• SELECT ROBOT
  • Favorites
  • Enter channel number
  • Scan (improved function)

• ROBOT
  • Start robot
  • Stop robot
  • Start charging
  • Stop charging
RemoteControl_GC.c / .h

• Interrupts routines
  • INT_RTCC : timer during robot control
    ➢ call to update function every 104ms
  • INT_EXT : ‘enter’ button
  • INT_EXT1 : ‘cancel’ button
  • INT_RB : ‘plus’ and ‘minus’ buttons
    ➢ call to press_button(char button)

• void main()
void main()

• Initialization sequence
  • Initialisation of the display
  • Synchronisation with the radio : `sync()`
  • Calibration of the joystick
  • Setting of the previous and current states
  • Enabling of the interrupts for the buttons

• Implementation of the state machine
  • Update the display according to the current state if necessary.
State machine

Changes between states:
void press_button(char button)

Actions during states:
void main()
Test for display update

« Update the display if the current state is different from the previous state, except if the current state is MENU and the display did not change »

\[(\text{current state} \neq \text{previous state}) \quad \text{XOR} \quad \text{[(current state} = \text{menu}) \quad \text{AND} \quad (\text{display}\_\text{changed} = 0)\]
Indication of charge (RC)

- Software implementation of the remote control charge indication using LEDs
- Indication only in RC_AUTONOMOUS state:
  current state = previous state = RC_AUTONOMOUS
- Current measurement (DS2764) :
signed 12 bits value

50 mA < I

30 mA < I < 51 mA

0 mA < I < 31 mA
Parts of the project

• **Hardware**
  • User interface
  • Battery management

• **Software**
  • User interface’s PIC18F2580

• **Housing and mechanical integration of the hardware**

✓ Dev. Version
✓ Final version
✓ Functions.c / .h
✓ RemoteControl_GC.c / .h
✓
Integration of the PCB

Remote control for CPG-based robots - Gabriel Cuendet
Elevation

• Choice of the buttons

⇒ Buttons height : 19 mm
- Choice of the buttons,
- Integration of the LCD
Elevation

- Choice of the buttons,
- Integration of the LCD and the battery

⇒ Height of the battery: ~8 mm
Ergonomics of the user interface

• Inspiration from the Logitech gamepad
Method

• Use of « Solidworks » for modelling
• Mechanical drawings of the two panels which were drilled (cf. report)
Result
Acknowledgements

• Alessandro Crespi

• André Guignard

• Mathieu Porez
Questions

Thank you for your attention