Using the *Rhoban ARM9 Plateform*

Hugo Gimbert Olivier Ly

February 3, 2011

1 Introduction

The *Rhoban ARM9 Plateform* is a set of tools which provide an easy way to quickly design and control robotic prototypes. The *Rhoban ARM9 Plateform* is suitable for prototypes driven by AT91 micro-controllers running Linux.

1.1 Features

The main feature of the *Rhoban ARM9 Plateform* is to provide easy access to and control of devices physically connected to the AT91 micro-controller. In particular, the *Rhoban ARM9 Plateform* allows:

- controlling generic digital output signals,
- sampling digital input signals,
- controlling servos and motors via pwm signals,
- controlling Dynamixel servos,
- sampling analog sensors,
- sampling I2C sensors,
- controlling cameras,
- controlling I2C and serial devices,
- etc...

1.2 Content

The *Rhoban ARM9 Plateform* consists in:

- a module which takes care of control and communication with the physical devices connected to an AT91 micro-controller,
- a C API which allows easy access to the devices from userland,
- a tool chain to compile C/C++ embedded programs,
- documentation,
- technical support from the *Rhoban* team ;-).

1.3 License

The *Rhoban ARM9 Plateform* was developped by Hugo Gimbert and Olivier Ly and is the property of its developpers.

Although the *Rhoban ARM9 Plateform* is proprietary software, it can be used freely for recreational and educational purposes.

Any commercial use is disallowed unless and end-user license agreement has been granted to the user for this particular use.

2 Quick Start Guide

Before starting working with the *Rhoban ARM9 Plateform* , the user should get familiar with Linux and C/C++ programming under Linux. This guide is applicable to version 20101220 of the *Rhoban ARM9 Plateform* . Version 20101220 is based on Linux kernel 2.6.

2.1 Using the C API control_low_level

This section provides a rough description of the low level API. More details are to be found in the header file *control_low_level.h*.

2.1.1 Initializing the low level system

First thing to do is initializing the low level system. This creates a communication link between the module and the API. This is done by a call to:
`int low_level_init(void);`

2.1.2 Listing and accessing devices

After having initialized the low level, the list of devices can be obtained in two ways. A pretty print of devices can be obtained using `void print_devices_infos(void)`. The complete list of devices and pointers to the devices themselves is stored in the `extern ModuleConnexion* etat_global` structure.

2.1.3 Writing outputs

A desirable basic feature is to control the state of the output pins of the controller. For each pin to control, load a digital writer device using:
`DigitalWriter * attach_digital_writer(ui32 frequency, ui32 pin, const char * name)` The frequency is the refresh frequency of the pin value by the module. The pin to be driven is specified by an integer, using the encodiing in *arch/arm/mach-at91/include/mach/gpio.h*. Name is an optional name to be given to the writer, and can be set to NULL.

To actually set an output, use `void set_digital_output(ui32 id, ui32 value)` .

Digital writers can be used to send sequences of pulses, which are specified by a sequence of durations and values, see the `Pulse` and `DigitalWriter` for more details.

2.1.4 Attaching a pwm device

Pwm devices can be used to generate pwm signals and thus directly control pwm servos.

To attach a pwm device use `PwmDevice * attach_pwm(ui32 frequency,ui32 pin, const char * pwm_name)` where frequency is the frequency of th epwm signal (50 Hz) for most devices. Only pins associated to a TIOA or TIOB signal can generate pwm signals this way. Each timer of the at91 microcontroller generates two signals TIOA and TIOB. Timer used by Linux (usually timers 0 and 1) should not be concurrently accessed by the module. On the at91sam9260 micro controller, pins usable to generate pwm signals are AT91_PIN_PB0 (TIOA3), AT91_PIN_PB1 (TIOB3), AT91_PIN_PB2 (TIOA4), AT91_PIN_PB3 (TIOA5), AT91_PIN_PB18 (TIOB4) and AT91_PIN_PB19 (TIOB5).

To set the cyclic rate of the signal, the signal should be first enabled and then the cyclic rate can be set using `void set_pwm_cyclic_rate (PwmDevice *, ui32 rate)`. The rate is expressed in thousands, for example at 50Hz (20ms) , a rate of 80 gives rise to high state during 20/1000*80=160 μs.

2.1.5 Reading sensors

Sensor devices include digital reader, magnetic sensor and analogic input. Each sensor has a refresh frequency. With each sensor device is associated the queue of the last 512 values measured at the sensor frequency.

2.1.6 Reading inputs

DigitalReaders are sensor devices that can be used to read the state of an input pin. Actually, not only digital readers read can be used to access the current value of a pin but they also monitor the sequence of state changes of an input pin.

To create a digital reader, use: `DigitalWriter * attach_digital_writer(ui32 frequency, ui32 pin, const char * name)`

2.1.7 Sampling an analogic input

AnalogicSensors are sensor devices that can be used to read analogic values.

To create a digital reader, use: `DigitalWriter * attach_digital_writer(ui32 frequency, ui32 pin, const char * name)`

On the at91sam9260 micro controller, there are four pins that can be used to measure analogic values are AT91_PIN_PC0 (AD0), AT91_PIN_PC1 (AD1), AT91_PIN_PC2 (AD2) AT91_PIN_PC3 (AD3).

2.1.8 Attaching a Dynamixel system

Also system of dynamixel servos can be controlled by the *Rhoban ARM9 Plateform* .