2013 International Conference on Robotics, Biomimetics, Intelligent Computational Systems (ROBIONETICS)
Yogyakarta, Indonesia, November 25-27, 2013

On Using the *Gameboy Advance* as a Controller for inverted Pendulums

D. Block, N. Gatzke and R.S. Sreenivas
Coordinated Science Laboratory, and
Industrial and Enterprise Systems Engineering
University of Illinois at Urbana-Champaign
Urbana, IL 61801
Corresponding Author's Email: rsree@illinois.edu

***Abstract*—This paper describes the usage of a *Gameboy Advance* (GBA) for non-gaming applications in control systems. The GBA is extended with the *XPort* and the *XPort Robot Controller* developed by Charmed Labs. Typical control problems include balancing an inverted pendulum about its unstable vertical equilibrium point. This paper also explores the capabilities of this controller, which can serve as a cheap, widely available microcontroller that can find use in college-level control systems classes.**

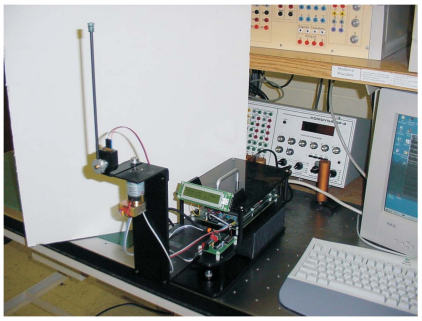
I. Introduction

The *Gameboy* by *Nintendo*, with the *Gameboy Advance* (GBA) in the third generation, is probably the most successful game console the world has seen. Despite its abilities in gaming, from an engineering point of view it can be seen as an embedded microcontroller, using an industry-standard processor accessible over the GBA's cartridge connector. It is a cheap and widely available low power system, supporting advanced graphics and sounds [1] The *XPort* by *Charmed Labs*[2] expands the abilities of the GBA by connecting it to a field-programmable gate array (FPGA), providing 64 general purpose input/output (I/O) signals that can be programmed to communicate with practically any kind of hardware. The XPort together with the additional robot daughter board forms the *XPort Robot Controller* (XRC). The daughter board provides in addition to the XPort other features like analog-to-digital conversion (ADC). This makes the GBA, together with the XRC, as an ideal controller for a variety of control problems.

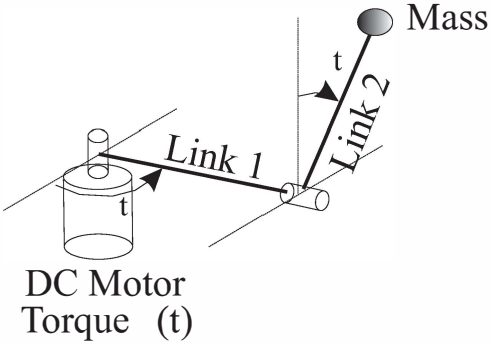
In this paper we present the relevant details of balancing an inverted pendulum with the aforementioned equipment. Specifically, we use the GPA and XRC to balance a *Furuta Pendulum*, which is a two link inverted pendulum used in control classes at the University of Illinois at Urbana-Champaign (cf. figure 1). The implementation of a controller of the Furuta Pendulum on the GBA in combination with the XPort is described in the remainder of the paper.

II. The Controller

The GBA in combination with the XPort and XRC offers a powerful embedded micro-controller that can be used in all different kinds of applications. As this combination was used to control two different systems during this project, a description of the device and its components is given in the following section. Also, a brief introduction on how to program the system is given.



(a) The Furuta Pendulum Set-up.



(b) The Furuta Pendulum Schematic

Fig. 1. The Furuta Pendulum.

A. Gameboy Advance and the Gameboy Advance SP

The GBA (cf. figure 2(a)) was first released in March 2001. In 2003, another version of the GBA was released, called the GBA Special Project (GBA SP; 2(b)). Apart from a lighted display, the new battery and the new design, the hardware remained the same as the 2001 version uses, thus the GBA as well as the GBA SP can be used to operate with the XPort4. A detailed description of the hardware the video game system includes is following, based on [1].

[1] An used version of GBA could be purchased for about US $30.00, and an used version of the GBA SP costs about $37.90 on Amazon.

[2] http://www.charmedlabs.com

(a) GBA. (b) GBA SP

Fig. 2. The GBA and GBA SP (Figure copied from www.nintendo.com).

The GBA's central processing unit (CPU) is an ARM7TDMI RISC chip running at 16.78MHz, supporting a 32-bit ARM mode as well as a 16-bit THUMB mode. For the XPort, a THUMB compiler is used, thus the processor runs in the 16-bit mode. This is justified by the fact that nearly all data used is received over the cartridge bus, which can be accessed only 16-bits at a time, and running in 32-bit ARM mode would degrade its performance. The memory map of the GBA is shown in figure 3. It can be seen that the different kinds of memory are accessed with a different width of data. Only the memory directly connected to the ARM CPU uses the full potential of 32 bits. The cartridge ROM, which is mainly used, is accessible with 16 bits a time.

The cartridge slot is originally used to connect the GBA to external memory, e.g. games burned on ROM on the cartridges. The pin assignment of the cartridge slot is shown in figure 4. The cartridge address bus is 24 bits width, providing access to a maximum of 32Mbytes of external memory. If the ROM is accessed, the 24-bit address is sent through pins AD0-AD23, then, 16 bits of data are transferred through pins AD0-15. This procedure is used to communicate with the 4Mbytes flash memory of the XPort.

Address Range	Purpose	Size	Data Width /[bits]
0000:0000-0000:3FFF	BIOS	16KB	32
0200:0000-0203:FFFF	External Work RAM	256KB	16
0300:0000-0300:7FFF	Internal Work RAM	32KB	32
0400:0000-0400:03FE	I/O Registers	1KB	32
0500:0000-0500:03FF	Background Palette RAM and Object Palette RAM	1KB	16
0600:0000-0617:FFFF	Video RAM	96KB	16
0700:0000-0700:03FF	Object Attribute Memory	1KB	16
0800:0000-variable	Cartridge ROM	max 32MB	16
0E00:0000-variable	Cartridge SRAM	max 64KB	8

Fig. 3. Memory map of the GBA.

The GBA has a 2.9 inch colored TFT LCD with a resolution

Pin	Name*	Dir**	Explanation
1	VCC	O	3.3V
2	PHI	O	System Clock
3	/WR	O	Write Select
4	/RD	O	Read Select
5	/CS	O	ROM Chip Select
6-21	AD0-AD15	I/O	Address Bus bits 1-16 or ROM Data bits 1-16
22-29	AD16-AD23	I/O	Address Bus bits 17-24 or SRAM Data bits 1-8
30	/CS2	O	SRAM Chip Select
31	/IRQ	I	Interrupt request
32	GND	O	Ground

* "/" indicates inverted signal

** Dir = Direction: I = Input, O = Output

Fig. 4. Pin assignment of the cartridge slot.

of 240 x 160 pixels at a maximum depth of 15 bits (32,768 colors). Each pixel information is stored in a 16-bit register, where each RGB color is represented by 5 bits (the 16th bit is not used). It supports up to 6 different display modes; briefly, they can be described as follows – Modes 0-2 are so called tile modes. This means, the background is not drawn pixel by pixel, but in blocks of 8x8 pixel sized tiles; every tile refers to a palette containing 256 colors. This has the advantage that a screen can be updated quite fast (these modes are e.g. used for displaying text). Here, a frame rate of 59fps can be reached. Modes 3-5 are bitmap modes. The background is drawn pixel by pixel. Using the full resolution of 240x160, it is not possible to achieve a frame rate of 59 fps. One solution is to reduce the frame rate to 30 fps. Alternatively, to keep the frame rate at the higher level, the resolution is changed to 160x128, or the color spectrum is reduced from 32,768 to a 256 colors containing palette. To display a picture in a bitmap mode, a vector in C has to be created, containing the 16-bit raw data for each drawn pixel. As a direct memory access (DMA) channel is used for copy- ing the picture data into VRAM with 32-bits at a time, the format of the vector entries should be 32-bit values, i.e. one vector entry should contain the data of two pixels

For playing sound, the GBA includes a Digital Signal Processor (DSP) supporting 2 DMA direct sound channels, but it has only one speaker (two sound signals can be mixed). A sound file is usually in 16KHz, 8-bit mono pulse code modulated (PCM) format, which has to be converted in raw data containing 32-bit entries. Similar to displaying the graphic, a DMA channel is used for copying the data into the right registers for playing sound. A timer is used to produce the aforementioned sound file.

The GBA provides four Timers, which can be run either at the CPU cycle frequency of 16.78MHz or a fraction of it. Each timer corresponds to a 16-bit count register, which can be configured such that if an overflow occurs an interrupt is called or the timer with the next higher index is started. Timer 0 and Timer 1 can be used to produce a sample frequency for

the sound system.

Four DMA channels are available, where the lowest number indicates the highest priority DMA. Channel 0 is used for highly time critical operations, whereas Channel 1 and 2?s main purpose is transferring the sound data. The external ROM (eg. the flash memory on the XPort) can be only be accessed through Channel 3, which is mainly used for copying image data.

The GBA provides 14 different interrupts. The important ones for this work include the vertical blank (V-Blank) interrupt, timer overflow interrupts and the external interrupt accessible over the cartridge. The last one is especially valuable to make interrupt calls possible over the XPort. Once an interrupt occurs, it is checked which one occurred and the corresponding interrupt service routine (ISR) is called, during which the interrupt has to be reset. An important fact is that if an interrupt occurs while an ISR called by another interrupt is executed, this ISR is exited.

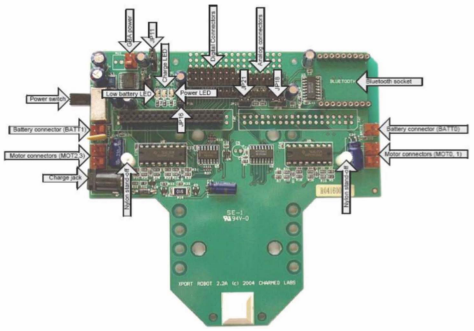
The Keypad provides a four key direction pad and six buttons, i.e. a total of ten buttons that can be used as a user interface.

B. The XPort Robot Controller

The XPort Robot Controller (XRC) consists of the XPort 2.0 (cf. figure 5(a)) and a daughter board, called "robot daughter board" (cf. figure 5(b)). The XPort is a board which connects to the GBA over the cartridge slot. Its center unit is a Spartan II field programmable gate array (FPGA) operating at 50MHz, including either 50,000 logic gates (standard version, Spartan II XC2S50) or 150,000 (extended version, Spartan II XC2S150). An overview of XPort 2.0 can be found in figure 6. The FPGA provides a total of 64 user programmable Input/Output (I/O) signals. The on board complex programmable logic device (CPLD) assists in the power-up configuration process that is required to program the FPGA when power is applied to the system. 4Mbytes of flash memory are available, which is used for saving the code executed by the GBA as well as the FPGA?s logic configuration. Communication between the XPort and a PC is provided by the on-board high-speed communication port (CPort), which is connected to a parallel port on PC side. It is used to program the flash (at a transfer rate of 50Kbytes/sec) and the FPGA, as well as for debugging. An optional 16 Mbytes of SDRAM can be added, which can be either addressed by the GBA or the XPort for saving and retrieving data. The most important task of the FPGA is demultiplexing. The GBA has a 16-bit cartridge design as opposed to the 8-bit design of the original Gameboy, but they share the same physical cartridge slot. To keep the number of cartridge connections and yet handle the extra amount of data, the GBA multiplexes all addresses and signals. However, to be able to communicate with the industry standard flash device on the XPort, the FPGA has to de-multiplex all those signals from the GBA side. In spite of this, several logic gates on the FPGA are available for custom configuration.

While the XPort is a strictly digital I/O device, the additional robot daughter card adds a spectrum of other possible

(a) XPort 2.0.



(b) Daughter Board

Fig. 5. The XPort 2.0 and the "robot daughter board."

applications. It includes four 1A H-Bridges for driving up to four motors and a 5V, 1A voltage regulator. A 12-bit analog-to-digital converter embedded in a circuitry is used for reading a back electro-magnetic field (back-EMF) signal to determine the velocity of connected motors. A second 12-bit ADC provides eight analog input pins, sitting right next to eight, 2-Channel general purpose I/O connections. A Bluetooth socket offers the possibility to easily add a module for wireless communication. An overview of the robot daughter card, connected to its peripherals, is given by the block diagram shown in figure 7. The FPGA is the central unit of the XPort. As it is part of the Xillinx chip family, which can be programmed in Verilog. Several modules for communicating with different kind of sensors, like e.g. optical encoders, are already included in the software package that is provided for the XPort by Charmed Labs.

As the GBA uses an industry standard CPU, the ARM7TDMI, different compilers are available to translate both C and C++. As some of the libraries of the XPort used are programmed in C++, this was the preferred compiler. Due to the scarcity of references on GBA programming, this project used the TONC project, originally written by T. Vijn [4]. The processor on the GBA is not able to handle floating point operations. Consequently, every part of the code has to be based on integer math. Additionally, there is no hardware unit included that is able to perform divisions. Therefore, any division operation is done by software emulation, which can result in long processing times. It is imperative that division be avoided for any time critical operation. One way to do that

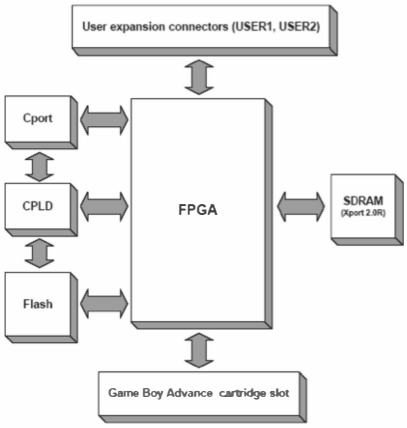


Fig. 6. Flow Diagram of the XPort [2].

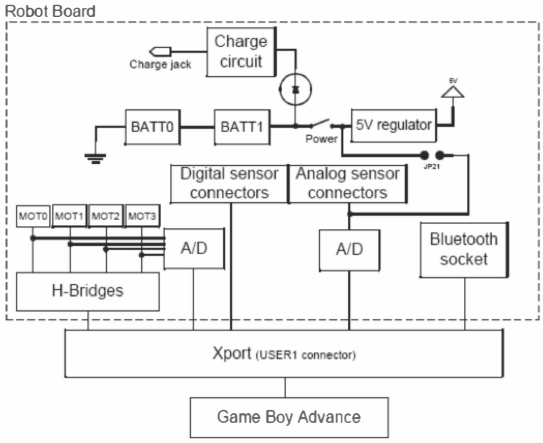


Fig. 7. Overview of the Robot Daughter Card [3].

is multiplying by the reciprocal. However, as floating points should be avoided as well, the reciprocal should be scaled, ideally by a factor with a power of two, because the re-scaling division can then be done by a bit-shifting operation saving several clock cycles. Finally, due to the multiplexing of the GBA, the addresses used by the GBA software defined in C/C++ differ from those defined on the XPort in Verilog. This means that the addresses from the FPGA have to be converted for the GBA. The developed code was tested with the help of a GBA emulator, the VisualBoyAdvance [5], so that the code did not have to be flashed on the XPort each time it was changed.

III. Furuta Pendulum

The *Furuta Pendulum* is a two-link inverted pendulum. Link 1 is directly connected to a motor, while link 2 connected to link 1. Link 2 is not actuated (i.e. it does not have a motor driving it). This is shown in figure 1(a), a typical laboratory set-up of the Furuta pendulum is shown in figure 1(b)[3].

[3]This apparatus was built by Dan Block, the manager of the *College of Engineering Control Systems Laboratory* (COECSL) at the University of Illinois at Urbana-Champaign.

The dynamic equations that describe the motion of this system are from Åkesson's thesis [6]. The physical parameters of the system are as follows:

$\tau(t)$: torque exerted by the motor on link 1,
l: length of link 2,
M: mass attached to the end of link 2,
m: mass of link 2,
J: moment of inertia of link 2 about its end,
J_p: moment of inertia of link 1 about its end, and
r: length of link 2.

$\theta_2(t)$ is assumed to be zero when link 2 is in the upright position, and it has a positive value when link 2 is moving in the clockwise direction with respect to the zero-reference. $\theta_1(t)$ is positive when link 2 rotates counter-clockwise to the zero-reference shown in the figure. The equations of motion are

$$\begin{aligned}
&\left(J_p + Ml^2\right)\left(\frac{d^2}{dt^2}\theta_2(t) - \left(\frac{d}{dt}\theta_1(t)\right)^2 \sin\theta_1(t)\cos\theta_1(t)\right) + \\
&Mrl\left(\frac{d^2}{dt^2}\theta_1(t)\right)\cos\theta_2(t) - gl\left(M + \frac{m}{2}\right)\sin\theta_2(t) = 0 \\
&Mrl\frac{d^2}{dt^2}\theta_2(t)\cos\theta_2(t) - Mrl\left(\frac{d^2}{dt^2}\theta_2(t)\right)^2 \sin\theta_2(t) + \\
&2(J_p + ml^2)\frac{d}{dt}\theta_1(t)\frac{d}{dt}\theta_2(t)\sin\theta_2(t)\cos\theta_2(t) + \\
&(J + mr^2 + Mr^2 + (J_p + ml^2)\sin^2\theta_2(t))\theta_1(t) = \tau(t)
\end{aligned}$$

Letting $a = J_p + Ml^2$, $b = J + Mr^2 + mr^2$, $c = Mrl$, and $d = lg(M + \frac{m}{2})$, the above equations can be written as

$$\begin{aligned}
&a\frac{d^2}{dt^2}\theta_2(t) - a\left(\frac{d}{dt}\theta_2(t)\right)^2 \sin\theta_2(t)\cos\theta_2(t) + \\
&c\frac{d^2}{dt^2}\theta_1(t)\cos\theta_2(t) - d\sin\theta_2(t) = 0 \\
&c\frac{d^2}{dt^2}\theta_2(t)\cos\theta_2(t) - \\
&c\left(\frac{d}{dt}\theta_2(t)\right)^2 \sin\theta_2(t) + 2a\frac{d}{dt}\theta_2(t)\frac{d}{dt}\theta_1(t)\sin\theta_2(t)\cos\theta_2(t) + \\
&(b + a\sin^2\theta_2(t))\frac{d^2}{dt^2}\theta_1(t) = \tau(t)
\end{aligned}$$

The state-vector is defined as $[\theta_1(t)\ \theta_2(t)\ \frac{d}{dt}\theta_1(t)\ \frac{d}{dt}\theta_2(t)]^T$, and the above set of equations are linearized about the state $[0\ 0\ \frac{d}{dt}\theta_1(t)\ 0]^T$, which results in the following linearized, state equations for the Furuta pendulum.

$$\frac{d}{dt}\begin{pmatrix}\theta_1(t)\\ \theta_2(t)\\ \frac{d}{dt}\theta_1(t)\\ \frac{d}{dt}\theta_2(t)\end{pmatrix} = \begin{pmatrix}0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & \alpha & 0\\ 0 & 0 & \beta & 0\end{pmatrix}\begin{pmatrix}\theta_1(t)\\ \theta_2(t)\\ \frac{d}{dt}\theta_1(t)\\ \frac{d}{dt}\theta_2(t)\end{pmatrix} + \begin{pmatrix}0\\ 0\\ \delta\\ \gamma\end{pmatrix}$$

where

$$\begin{aligned}
\alpha &= \frac{ab\left(\frac{d}{dt}\theta_1(t)\mid_{t=0}\right)^2 + bd}{ab - c^2} \\
\beta &= \frac{-ac\left(\frac{d}{dt}\theta_1(t)\mid_{t=0}\right)^2 - cd}{ab - c^2} \\
\gamma &= \frac{-c}{ab - c^2} \\
\delta &= \frac{a}{ab - c^2}
\end{aligned}$$

The Furuta pendulum used in this project has the following state-equation (assuming $\frac{d}{dt}\theta_1(t)\mid_{t=0}= 0$):

$$\frac{d}{dt}\begin{pmatrix}\theta_1(t)\\ \theta_2(t)\\ \frac{d}{dt}\theta_1(t)\\ \frac{d}{dt}\theta_2(t)\end{pmatrix} = \underbrace{\begin{pmatrix}0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ 0 & -52.06 & 0 & 0\\ 0 & 76.18 & 0 & 0\end{pmatrix}}_{\mathbf{A}}\begin{pmatrix}\theta_1(t)\\ \theta_2(t)\\ \frac{d}{dt}\theta_1(t)\\ \frac{d}{dt}\theta_2(t)\end{pmatrix} + \underbrace{\begin{pmatrix}0\\ 0\\ 34.41\\ -13.35\end{pmatrix}}_{\mathbf{B}}\tau(t)$$

Also, there are optical-encoders on this pendulum that provide the values of $\theta_1(t)$ and $\theta_2(t)$ directly. The values of $\frac{d}{dt}\theta_1(t)$ and $\frac{d}{dt}\theta_2(t)$ will have to be estimated from these observations. As a consequence we have the output equation for the system to be

$$\mathbf{y}(t) = \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\end{pmatrix}\begin{pmatrix}\theta_1(t)\\ \theta_2(t)\\ \frac{d}{dt}\theta_1(t)\\ \frac{d}{dt}\theta_2(t)\end{pmatrix}$$

A. Pole Placement Design

We first look at the eigenvalues of the **A** matrix for stability. The system is unstable as there is an eigenvalue on the right-hand-side of the s-plane. Our goal to keep link 2 in the balanced position for as long as necessary. The system is controllable and observable, so we can place the closed-loop system's poles anywhere we like, and we can build observers for $\frac{d}{dt}\theta_1(t)$ and $\frac{d}{dt}\theta_2(t)$. For instance, if we were to place the closed-loop system's poles at the following locations $\mathbf{P} = [-9\ \ -10\ \ -11\ \ -12]^T$ (these choices are ad-hoc; but the resulting system will be stable for these choices). This results in the following gain matrix

$$\mathbf{K} = \begin{pmatrix}-6.1671 & -70.9516 & -2.3765 & -9.2702\end{pmatrix}.$$

We had access to all state-variables, the above gain will accomplish our objective. But, we only have access to $\theta_i(t)(i = 1, 2)$[4], we need observers for $\frac{d}{dt}\theta_i(t)(i = 1, 2)$, one possibility is to build the following naive observer

$$\widehat{\frac{d}{dt}\theta_i(t)} = \frac{\theta_i(t) - \theta_i(t-T)}{T}(i = 1, 2),$$

for a reasonably small value of T. One could also build a reduced order observer for this system if necessary.

Attempts at getting the control algorithm on the GBA with floating point operations were unsuccessful. The time duration for the four calculations for the control effort of each state takes more than 3ms with floating point operations, while the integer based one took only about 100?s. Consequently. the floating point operations had to be changed to an integer math base. A timer is configured in the code to call an interrupt, and consequently the control function, every 5ms. The input of the encoders is used to determine the angle position of each link as well as each velocity. Those four states are then multiplied by their proportional gain. A check occurs if the user already activated the control. If this is true, the calculated output is sent to the motor if the Furuta Pendulum is not out of its operation range. The control algorithm itself is illustrated in the flow graph shown in figure 8.

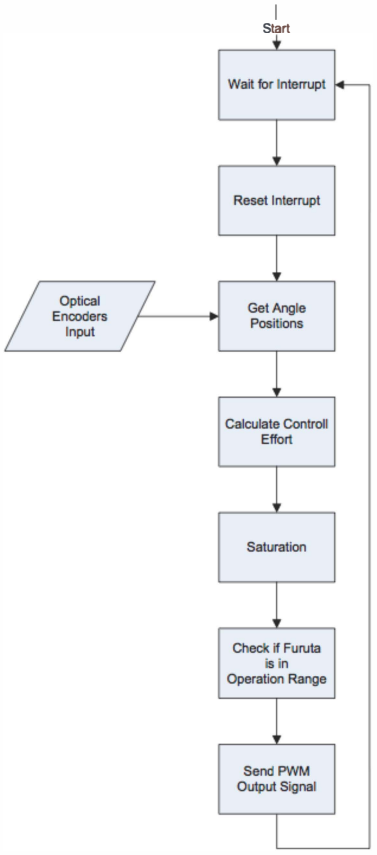


Fig. 8. Flow diagram of the control algorithm.

Once the Gamboy is started, it will show a start screen that can be passed by pressing the Start button. Following this the menu shown in figure 9 appears on the screen. Once the Start button is pressed, the PWM output is active, and the Furuta Pendulum will balance. The left and right button can be used to change the reference angle for θ_1, thus the base of the pendulum can be moved while it keeps balancing. In addition, with the A and B button quick changes of the reference angle for θ_1 can be made. The figure 11 is a screenshot of the menu. A picture of the Furuta Pendulum controlled by the GBA in addition with the XPort is shown in figure **??**:

IV. Conclusion

The Furuta Pendulum was successfully balanced by the GBA in combination with the XPort. One of the main motivation of the project was to use the results as a teaching aid in

[4]Two optical encoders (S1-1000-B from US Digital, http://www.usdigital.com) are mounted to measure the angle position of both links. One 12V DC motor (Pittman 8222, http://www.pennmotion.com) is connected to link 1, with a stall torque of 7.4 oz*in.

Fig. 9. Welcome screen of the Furuta Pendulum control.

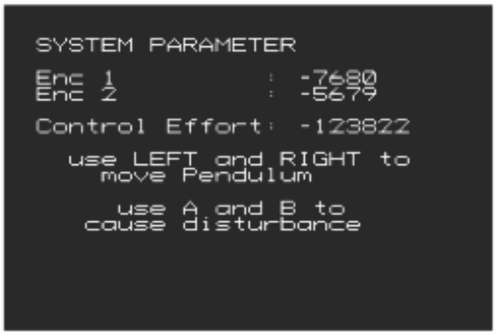


Fig. 10. Welcome screen of the Furuta Pendulum control.

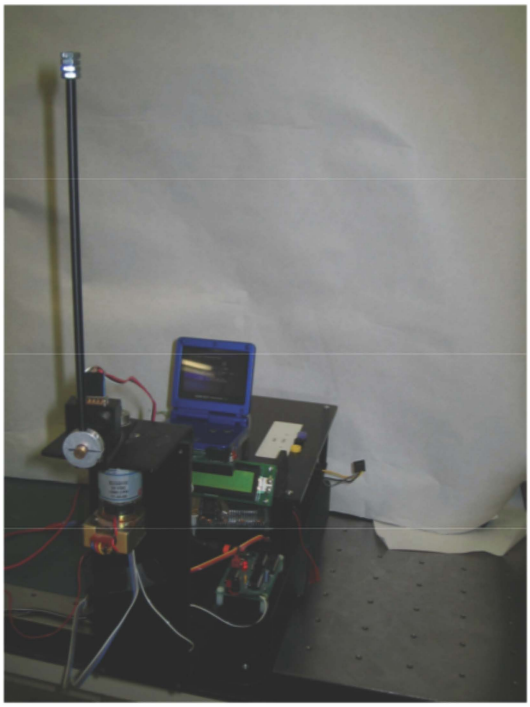


Fig. 11. Welcome screen of the Furuta Pendulum control.

control systems classes at the University of Illinois at Urbana Champaign. The main drawback to this was the fact that the GBA's inability to perform floating point math operations as well as divisions with an acceptable time. The scaling that is necessary to avoid these operations resulted in complex code that obfuscated the pedagogical motivation for this project. As a result this system has found limited use in our classroom.

REFERENCES

[1] "Gbatek," URL http://nocash.emubase.de/gbatek.htm.
[2] "Xport 2.0 user guide v.1.5," Charmed Labs, http://www.charmedlabs.com.
[3] "Xport robot controller user guide v.1.1," Charmed Labs, http://www.charmedlabs.com.
[4] "Tonc v1.4.2," URL http://www.coranac.com/tonc/text/toc.htm.
[5] "Visual boy advance v.1.5.1," URL http://vba.ngemu.com.
[6] J. Åkesson, "Safe manual control of unstable systems," Technical Report: ISRN LUTFD2/TFRT-5646-SE, Department of Automatic Control, Lund Institute of Technology, Sweden, September 2000.