# STANDALONE GAMING CONSOLE USING ARDUINO

DEOVRAT PHAL, ADITYA S, AKSHAY NARKAR, RAHUL GHUGE

**Abstract:** Multimedia games are everyone's favorite. Whether it be a MS-DOS based Warlords II game or Windows based Need for speed or even the latest craze, the Android based – Temple run or Subway Surfers, we all have at one point or another dreamt about personalizing and tampering with the console to create our very own gaming experience. Toying around with this thought, we decided to finally implement a gaming system with elementary concepts of open-source embedded gaming platform. In the paper we have presented a system developed using C programming on Arduino IDE. Our minimalistic embedded hardware platform includes Atmega328 microcontroller, 5-way Joystick and Graphic LCD Screen. The code for particular game is to be downloaded on to the controller board. The game development for single player PONG game has been discussed in the paper.

**Keywords:** Arduino, Graphic LCD, 5-way Joystick.

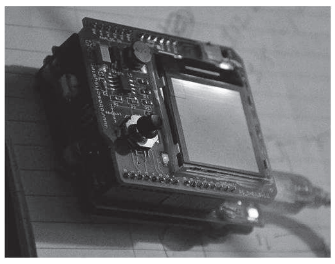
**Introduction:** Embedded systems have a vast amount of useful applications throughout the world. This has provided a new fillip to many technical enthusiasts for exploring the amazing field of embedded technology. Tailor-made features are incorporated carefully by programming the system. The video games which we play on TV also comprise of embedded systems that are programmed a priority to improve the user's real- time interaction with the system. The controllers of Sony PlayStations, Microsoft X-box, and Nintendo all have sophisticated embedded systems running them. This motivated us to develop our very own gaming system by perusing various bibliographic resources-both electronic and otherwise as also guidance from various knowledgeable personnel. Developing small embedded system presents several challenges, due to size and weight constraints. Size reduction implies the use of small microprocessors with limited memory and processing power, which represents a severe constraint for the software, which must be carefully designed to be efficiently exploiting the available resources. Moreover, if the system is powered by batteries, energy management policies must be applied at different levels of the architecture to reduce power consumption and prolong the system lifetime as much as possible.

PONG is one of the earliest arcade video games; it is a tennis sports game featuring simple two-dimensional graphics. The aim is to defeat the opponent in a simulated table tennis game by earning a higher score. The game was originally manufactured by Atari Incorporated (Atari), who released it in 1972. Allan Alcorn created Pong as a training exercise assigned to him by Atari co-founder Nolan Bushnell. Bushnell based the idea on an electronic ping-pong game included in the Magnavox Odyssey, which later resulted in a lawsuit against Atari. Surprised by the quality of Alcorn's work, Bushnell and Dabney decided to manufacture the game. The game has been remade on numerous home and portable platforms following its release. We have specifically focused on the PONG game development using ARDUINO Duemilanove and a Nokia 6110 Graphic LCD screen.

This system is based on Arduino platform. Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. The hardware used is an Arduino Duemilanove microcontroller board and a LCD shield. The software used is Arduino IDE. For advanced gaming features chipmunk's physics engine is used which has been optimized for giving real physics functionality to game objects.

**Hardware System:**

Fig 1: The gaming console



The controller board is illustrated in figure. It performs interfacing functions with the game; it provides us with joystick by which we can control the game.

***2.1 Controller Board:*** The controller board is Arduino Duemilanove, which consists of ATmega328 microcontroller. It is 8 bit microcontroller. The Arduino Duemilanove can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Ground and Supply pin headers of the POWER connector. The board can operate on an external supply of 6 to 20 volts. The recommended range is 7 to 12 volts. [4]

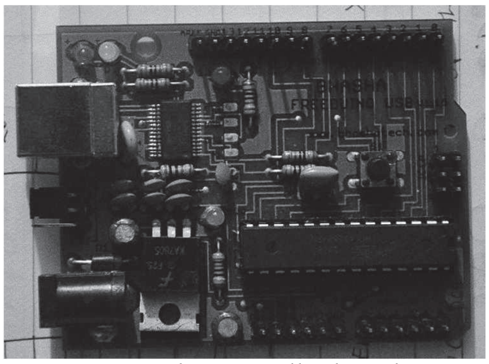


Fig. 2: The controller board

***2.2 Advanced RISC Architecture:*** It has 131 Powerful Instructions, most of them having a single clock cycle execution. It is also packed with 32 x 8 General Purpose Working Registers as well as Peripheral Control Registers. It has fully static Operation up to 20 MIPS Throughput at 20 MHz. It has on-chip 2-cycle Multiplier. [3]

***2.3 Peripheral Components:*** It has Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes. Also available is 16-bit Timer/Counters with Separate Prescaler, Compare Mode, and Capture Mode. It also has a Real Time Counter with a Separate Oscillator. It has six PWM output pins with Byte-oriented Two-wire Serial Interface. Programmable Serial USARTs and Master/Slave SPI Serial Interface with Programmable Watchdog Timer with On-chip Oscillator and On-chip Analog Comparator are additional attractive components. [3]

***2.4 Special microcontroller Features:*** Some of these are - Power-on Reset, Programmable Brown-out Detection, internal calibrated oscillator, Internal Interrupt Sources with Five Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, and Standby. It has Software Selectable Clock Frequency. Also Byte-oriented Two-wire Serial Interface and Dual Programmable Serial USARTs and Master/Slave SPI Serial Interface.Programmable Watchdog Timer with On-chip Oscillator and On-chip Analog Comparator are additional provisions. [3]

***2.5 SPI:*** There are two programs given for SPI, one for Master and another for Slave. You can communicate between two Arduino boards by connecting wires on SPI lines and making ground common of both the boards. The Output is debugged using the serial terminal whatever character typed on master's serial terminal is transferred to the slave's serial terminal using SPI.

The Color LCD Shield is a peripheral attached to the Arduino Duemilanove microcontroller board. It consists of a LCD screen and a Joystick (analog joystick of PS2 fame). The LCD screen is based on Philips PCF8833 driver. The screen has 132*132 pixels; each one with 12-bit color (4 bits RED, 4 bits GREEN and 4 bits BLUE), thus providing 4kB colors. RGB value of each pixel of LCD is programmable. It also has a Two-wire serial SPI interface for clock and data. The Joystick is used as a mouse on the LCD screen in this sample example. The Joystick X axis is connected to the on-chip ADC channel 1 and Y axis to channel 2. The ADC is configured such that it gives an interrupt after the conversion is over. Since only one ADC channel conversion is done at a time so we need to reconfigure it for other channel. The ADC conversion value is stored in ADC's data register. [6]

***2.6 USB programme :*** USB programmer is available on board which is used to embed the program in the microcontroller and also for providing power.

**Software System:**

***3.1 Arduino IDE:*** The game is coded using Arduino. Arduino IDE is a cross-platform application written in Java, and is derived from the IDE for the Processing programming language and the Wiring projects. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. A program or code written for Arduino is called a "sketch". Arduino programs are written in C or C++. [5]

***3.2 Code written for controlling game:*** We have to configure the board according to our game and write the appropriate code. Arduino programs are written in C or C++. Users only need define two functions to make a runnable cyclic executive program:

- setup(): a function run once at the start of a program that can initialize settings.
- loop(): a function called repeatedly until the board powers off.

If any interrupts are required for example: if have to exit a game then we can directly press the button which will interrupt the game and exit. We write the code in Arduino IDE software. The code is written in such a way that according to the movement of joystick, we get the corresponding values on LCD screen through which we are able to know whether proper synchronizing between the game control and our control joystick has been done. [4]

**Game Development:**

***4.1 Defining Constants:*** Set the maximum points for the game to 500. It can be any value, but care must be taken so as the memory in use is not over flown, which may result in stack error.

***4.2 Movement:*** Set the frame delay to value equal to

100ms. It is the animation frame duration in milli-seconds, and 100 is pretty neat value. Set the minimum ball speed to 800. So each round, ball starts at this speed. Set the maximum ball speed to 300. Once this speed is reached, the ball won't accelerate past this. It is advisable not to set high speeds as it may give rise to unknown bugs. To make game more interesting, we added a parameter called ball acceleration. We kept it at 50. It shows how much ball will accelerate at each bounce. The paddle speed should not be too fast or too slow. The value 130 is fine. The two factors namely X_FACTOR and Y_FACTOR increase the ball's dx/dy, i.e. making it move. To make ball move more horizontally and less vertically when it bounces, X_FACTOR is used to calculate a random variance in the resulting ball vector. Lower values increases dx/dy. 100 seems to work well for X_FACTOR. When the ball hits a paddle, its new vector is the line connecting the center of the paddle to the center of the ball. X value of this vector is multiplied Y_FACTOR. Higher Y_FACTOR increases the ball's dx/dy after hitting a paddle. 200 seem to work well for Y_FACTOR. Y_FACTOR is used to fix a bug where ball would get 'stuck' bouncing back and forth along the top or bottom wall. A collision with top or bottom wall causes a bounce where new dx is -(old dx). If old dx is 0 then new dx is 0 so ball never leaves wall. Y_FACTOR is added to dx when ball bounces off top or bottom wall. It should be small.

***4.3 Layout/structure:*** Layout and structure involves setting the parameters like ball radius, Wall width, Game field dimensions, Paddle dimensions. To make game more interesting we managed the appearances. For example we set the ball color to red, ball outline to brown, Ball shape to circle and paddle color to transparent with black outline.

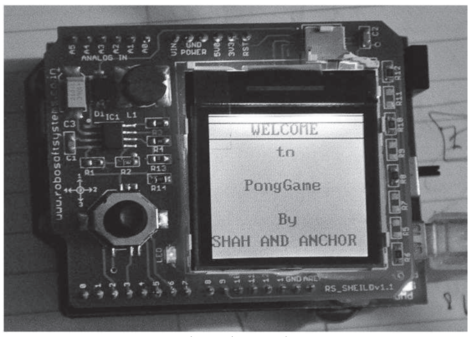


Fig. 3: Message displayed on screen

***4.4 Messages to be displayed:*** We first set the center location for messages to be displayed by setting the corresponding X and Y coordinates. When the game is initialized following text appears on the LCD screen 'Welcome to PongGame by SHAH AND ANCHOR". When the game is paused, following text appears on the LCD screen 'Game paused. Your current score is ***. Press middle button to quit game. Press any direction key to continue'.

***4.5 Variables used in PONG Game:*** Using Arduino IDE software we can design awesome games for the system. The game is developed by programming each pixel of the LCD screen. Likewise there is a predefined library of LCD Shield. These can be readily implemented by us. There are various codes as per specific functions. Some of the methods are described below:
startNew(), restart(), contrast(), setPixel(), setCircle(), BallVector, BallSpeed, ballX, ballY, setChar(), setStr(), setLine(), setRect(), paddle, drawFrame(), sliderMove(), Paused, Quit Game (It is Guard for main loop. When true, program ends)

***4.6. Subroutines:***

*4.6.1 Create Figure:* Create figure sets up the main game field. It plots the ball (circle to draw on the game field), walls and the paddle. It is called once at the start of the program.

*4.6.2 New Game:* When new game is called the game is reset to starting conditions. This function is called from main loop at program start. It is called when the user ends the game or wish to quit the existing game. At the start of the new game the score, paddle velocity, ball speed are set to value zero.

*4.6.3 Restart Game:* This function resets ball location, speed and direction. It is called from the function Newgame or after each point is scored.

*4.6.4 Move Ball:* When the ball hits either of the following: top boundary, right boundary, left boundary or the paddle, the X and Y coordinates of the ball are refreshed and the program calculates new ball location. For each hit the bounce is called to change ball vector.

*4.6.5 Move Paddle:* The paddle velocity has been predefined while burning the code into the console. To move paddle the X coordinates are constantly refreshed. The program written also ensures that paddle remains in the game field all the time.

*4.6.6 Bounce"* During the game play, the normalization vector sets as new ball vector. The game has been developed such that at each bounce, the ball accelerates, making it difficult for the user score points.

Logic for ball acceleration:

```
If (ballSpeed + BALL_ACCELERATION < MAX_BALL_SPEED)
    ballSpeed = ballSpeed + BALL_ACCELERATION;
```

Other subroutines include count points, pause game and unpause game.

***4.7 Main Script:*** The function of main script is to display the initialization text, start new game, refresh plot, move ball and paddle, count points.

**Conclusion:** The process of actual implementation of the project has given us ample exposure to the various advancements in the field of embedded gaming. We have successfully implemented a low-cost, low-power embedded gaming system at the heart of which is an AVR microcontroller. From the very basic design of gaming controllers to the actual synchronization between the game and the controller, every step has strengthened our basic concepts and enhanced our programming skills.

**Future Scope:** For advanced gaming features an accelerometer can be interface to the Arduino MCU, which will do the job of joystick. Further scope for development includes employment of HMI interface akin to Microsoft Kinect, Nintendo Wii, Wi-Fi enabled gaming. Augmented reality, which is still in the nascent stage of its development, may also be incorporated like in Nintendo 3DS and PlayStation Vita to enable gaming enthusiasts to have a real-life feel whilst playing.

**Acknowledgment:** The authors would like to thank the Shah and Anchor professor, DR. Uma Rao for her valuable support and feedback provided during the development of the project.

**References:**

1. J.Turley, "Embedded processors", http://www.extremetech.com , January 2002
2. Manzano, M; Hernandes. J A, "An empirical study of cloud gaming", 2012.
3. Kotsia I et.al. "Affective gaming: Comphrensive study", 1992
4. Sushant Sakhalkar, Parth Gala et.al "Embedded Gaming", IJERA, 2012
5. Simon Monk, "Programming Arduino Getting Started with Sketches, McGraw-Hill, November 2011.
6. Robosoft systems graphic LCD board.

***

(BE) Electronics, Mumbai ,p.deovrat@gmail.com,
(BE) Electronics, aditya.gsr2014@gmail.com,
(BE) Electronics, rahulghuge12@gmail.com
Shah & Anchor Kutchhi Engineering College, Mumbai,
(BE) Mechanical, Lokmanya Tilak College of Engineering, Mumbai,
akshaynarkar@gmail.com,