Wireless Multiplayer Retro Gaming Console

Zlate Bogoevski, Kristijan Nelkovski, Danijela Efnusheva and Marija Kalender
Computer Technologies and Engineering Department, Faculty of Electrical Engineering and Infomation Technologies, "SS. Cyril and Methodius University" in Skopje, Rugjer Boshkovikj Str. 18, Skopje, N. Macedonia
{zlateb,ksiar212018,danijela,marijaka}@feit.ukim.edu.mk

Keywords: IoT System Design, Wireless Communication, Video Game Hardware, Handheld Gaming, Motion Sensor, PONG.

Abstract: Handheld video gaming has been a staple in the video game industry for over thirty years. While the current generation of handheld video game consoles are much more powerful than their predecessors, a substantial community of enthusiasts still enjoy the tactile feel and retro aesthetics of much older and more traditional hardware and media. Newer handheld devices that aim to replicate this through software emulation and contemporary hardware typically lack the wireless multiplayer connectivity that has since become a key feature for modern day handheld and mobile video game titles and devices. In this paper, we analyze these technologies and their software by developing our own basic handheld gaming console with wireless multiplayer capabilities for cooperative and competitive gaming between devices through the use of Wi-Fi ESP8266 based microcontrollers. The results demonstrate the development of two separate nodes capable of complete wireless interconnectivity for simultaneously running a proof of concept multiplayer recreation of the classic "Pong" arcade game.

1 INTRODUCTION

While processor and microcontroller architectures have seen a substantial increase in their performance and capabilities in conjunction with advancements in miniature display technologies such as color, pixel density and low power consumption, companies and engineers have been working on a multitude of new types of handheld gaming devices, combining the possibilities offered by electronic and electrical engineering with the art and format of graphic and interactive storytelling offered by the video game medium.

Although the current generation of handheld gaming consoles such as the Nintendo Switch, the Steam Deck, and even mobile Android devices (both regular smartphones as well as dedicated consoles) are vastly superior to the technologies of the eighties and nineties, many video game enthusiasts are still interested in older generations of video games that they remember from their own past.

What perhaps started as childhood nostalgia has been cultivated by consumer electronics companies through the phenomenon known as "retro handheld" devices which have seen increasingly more powerful electronics capable of emulating a wide range of gaming systems from the earliest examples such as 1980s consoles like the Atari 2600, to some more recent systems like the early 2000s PlayStation 2, typically in the form factor, look and feel of some of the most popular handheld devices of the past, most notably Nintendo's GameBoy line of products (original GameBoy, GameBoy Color, GameBoy Advance).

The rapid development of these modern emulation machines has been made possible in part thanks to the widespread availability and affordability of microcontroller and single board computer platforms such as the Arduino and the Raspberry Pi, that have enabled software emulation of older video games, but have also allowed for developers to make recreations (clones) as well as their own custom titles with similar gameplay and art styles without having to interact with any legacy hardware and software.

The aim of this paper is to design a retro handheld gaming device complete with wireless networked multiplayer features capable of interacting with multiple nodes of the same kind.

The paper is organized as follows: Section 1 provides an Introduction. Section 2 analyses the state of the art. Section 3 gives details about the methodology used in the design of the proposed solution. Section 4 discusses the practical implementation and results. The paper concludes with Section 5.

2 STATE OF THE ART

The proposed project falls into the second category mentioned above, prompted by devices such as the Arduboy and Makerbuino which are standalone 32 bit AVR microcontroller based single board miniature gaming consoles capable of running classic arcade style games in the form of open source code offered by the manufacturers of these devices as well as community software later developed by the end users themselves.

Using wireless networked microcontrollers our goal was creating a similar type of device that was capable of not just running custom-made arcade style video games, but also have on board wireless connectivity, capable of interacting with other units of the same type allowing for both competitive and cooperative gameplay between multiple users.

Wired multiplayer has been enabled with traditional handheld video game devices even on consoles such as the classic Nintendo GameBoy, although nothing compared to the wireless Internet capabilities of the Nintendo Switch and the Steam Deck, of course in conjunction with the server infrastructure that their owner companies possess today.

This work was prompted by research into the technologies behind the classic GameBoy, notably two works that aim to bring wireless capabilities to this thirty plus year old device.

The first work that inspired our research is an ESP8266 based wireless GameBoy cartridge [1] developed by physicist Dr. rer. nat. Sebastien Staacks capable of connecting and browsing the Internet while not necessarily allowing for multiplayer gameplay on the device right out of the box. This device fits the form of a typical GameBoy game cartridge, traditionally used to load individual games onto the device, allowing for the user to develop software clients for browsing the Internet and graphically displaying this information using the device itself as an interface.

The second project that inspired our work is a RaspberryPi Pico wireless gaming link cable adapter [2] for the GameBoy, developed by IT security researcher Thomas Roth which does enable online multiplayer through manipulating the GameBoy link cable protocol via the RaspberryPi Pico.

While these two projects build upon the existing hardware of old and hard to come by technologies, as well as legacy and unstandardized software, our goal was to develop both a hardware and software solution that reflects the modern IoT and maker scene while being both affordable and capable of performing our desired task.

We are further referencing two more projects that have aided us in the development of a classic Pong game as a prototype/proof-of-concept for our device although vastly differing in the hardware and later software of our final work. These two projects are the Arduino Pong game developed from scratch by Mark Geddes [3] without the use of graphical display libraries created for teaching and understanding Arduino display driving via SPI (serial peripheral interface) and the How to write a game instructable project by Hari Wiguna [4] developed as a non-networked basic multiplayer Pong game on a single Arduino based breadboard device.

3 METHODOLOGY

This system is developed using an ESP8266 which is a low cost 32-bit Wi-Fi microcontroller based on the Tensilica Diamond Standard 106 Micro CPU architecture with 32 KB of instruction memory and 80 KB user data memory supporting up to 16 MB of external flash while having no internal ROM. This module features integrated TCP/IP networking software as well as other communications protocols such as SPI and I2C/UART. This device can perform as either a standalone unit with 17 dedicated general-purpose input output pins or act as a slave device for a host chip, reducing communications overhead for a main applications processor.

For this project the ESP8266 chip is used as a main independent microcontroller in the form of the commercially available WeMos D1 mini development board which includes 11 digital GPIO pins and the full Wi-Fi IoT capabilities of the ESP8266 platform (Figure 1). Although there are many vendors of ESP8266 microcontroller boards, the WeMos D1 mini is a widely used product with extensive documentation, a small footprint as well as convenient breakout pin headers, onboard 3.3v voltage regulation through the ME6211 chip and integrated USB to serial programming capabilities through the CH340 USB bus converter IC.

Individual nodes in our system are built with identical hardware and are able to facilitate fast full-duplex peer-to-peer communication via Espressif's ESP-NOW protocol which is based on the IEEE 802.11 b/g/n Wi-Fi wireless communication network standard and works on a radio frequency of 2.4GHz. Using this protocol our nodes can wirelessly transfer game information between multiple player devices and show correct positions, scores and flags to each individual device/player with extremely low latency without the necessity of a dedicated server or interfacing with the Internet.

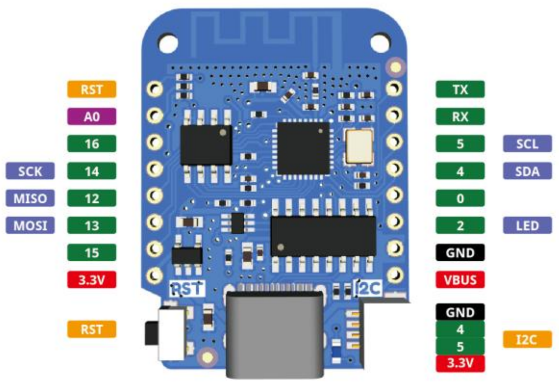

Figure 1: The WeMos D1 mini board and pinout.

3.1 Hardware

Each node in the proposed system is a standalone device created with a custom printed circuit board (Figure 2, Figure 3) and equipped with its own D1 mini board, a Nokia 5110 84 by 48-pixel graphic LCD, an ATtiny85 microcontroller, two piezoelectric buzzers, a GY-271 magnetic field sensor, a PCF8574 I2C remote 8-bit I/O expander, eight momentary input switches and two 2xAAA battery packs.

The Nokia 5110 LCD is a basic dot matrix graphic LCD display that uses the PCD8544 low powered CMOS controller driver which enables the control of 48 rows and 84 columns of pixels via SPI. This interface method offers synchronous serial communication for short distances primarily developed for embedded systems and extensively covered for the use and connectivity in numerous microcontroller platforms such as the ESP8266 boards. This display can only show 1-bit colors per pixel, has a diagonal of about 1.5 inches and features four LEDs as backlight, although not used for this system due to power efficiency.

Our system utilizes a singular ATtiny85 microcontroller as a simple solution for audio output processing. The ATtiny85 microcontroller developed by Atmel (now part of Microchip) is an 8-bit AVR RISC architecture device with 8 KB of ISP flash memory, 512 B of EEPROM memory and 512 B of SRAM memory. This module features six GPIO pins and has a clock speed of 8 MHz. In this project the ATtiny85 microcontroller is used to store multiple segments of MIDI (musical instrument digital interface standard) music and output it as background game music via a single piezoelectric buzzer.

This music is converted from original MIDI files into program code via Extramaster’s online “MIDI to Arduino” tool and is stored within the software files of the ATtiny program capable of being activated by the ESP8266 board through two pins for simplex communication (ESP to ATtiny only) and a third pin for exiting the music loop via resetting the program on the ATtiny85 chip. The ATtiny code can have multiple user defined loops of music which contain different MIDI snippets and can be chosen through the one-way communication by the ESP8266 chip.

While the D1 mini is also capable of performing the same function of MIDI playback through a piezoelectric buzzer, the ESP8266 device integrates only a single core processor that makes it difficult to run both software loops of musical notes as well as game and wireless communications code concurrently. While this can be done through the use of multithreading or via connecting different libraries and overcomplicating the main system code which might impair speed and system memory, using a dedicated microcontroller for background game music is a simpler solution without overwhelming the software running on the main ESP8266 processor.

For singular tones of game sounds which require only one line of code per tone for playback however, this does not represent any problems which is why our system employs a second piezoelectric buzzer that is directly controlled by the ESP8266 chip and can activate singular notes at a time whenever a game function requires it.

One of the two control input methods for our system is the GY-271 magnetic field sensor. This device is a three-axis magnetometer module that uses the Honeywell HMC5883L digital compass IC. The chip itself includes multiple state of the art high resolution HMC118X series magneto-resistive sensors with a 12-bit ADC that enables compass heading accuracy within one to two plane angle degrees. For communications this chip uses the I2C protocol and is intended for use with consumer electronics devices such as phones, tablets and vehicles, while also offering sufficient documentation for the use with microcontrollers such as the ESP8266. In this system this sensor is used as a motion controller for the player through the detection of the change in a user’s heading. This gives players the option to move their on-screen characters via their own physical movements in the real world.

The second input method for controlling our system are eight standard momentary switch push-buttons with a clicky tactile feel laid out in two thumb operated directional pads with one button on each point similar to most modern and retro gaming console gamepad and controller layouts. The layout of the buttons is as follows: the left directional pad is marked with left, right, up and down arrows while the right directional pad is marked with the first four letters of the Cyrillic alphabet: A, B, V and G. These buttons are meant to be used as the primary method of interaction and selection within the main system menu as well as subsequent game menus. Furthermore, the buttons can be used as the primary method for moving the on-screen game characters or as a secondary method for movement combined with the use of the previously mentioned GY-2711 magnetometer.

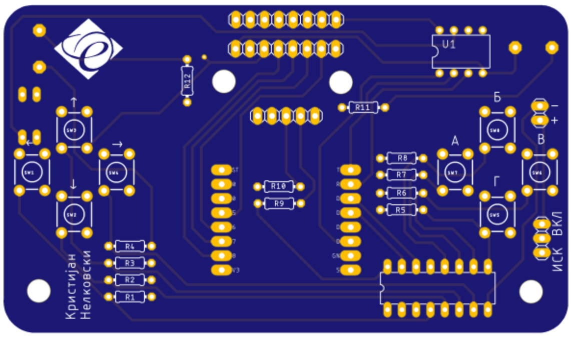

Figure 2: A CAD model of the front of the printed circuit board design.

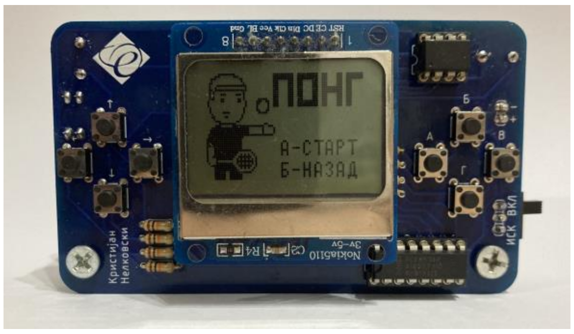

Figure 3: A powered on fully soldered, functional node.

Because the WeMos D1 mini only has 11 digital general-purpose input/output pins, in order to facilitate the stable interconnectivity with all of the previously covered hardware, the system utilizes a GPIO expander IC for reading out the eight button inputs. This is done using the Texas Instruments PCF8574 8-bit input/output expander connected to the ESP8266 through the same I2C bus as the GY-271 sensor. Identifying each device is done through software via a fixed address for the sensor and a three-bit user adjustable (by connecting these bits to either ground or 3.3v) hardware address input for the GPIO expander.

3.2 Software

The WeMos D1 mini board can be programmed either using the Arduino C/C++ programming language or the microcontroller MicroPython implementation of the Python programming language. For our system we chose to use the Arduino C/C++ language because of the multitude of up to date, heavily documented and readily available libraries that are compatible with each other and capable of connecting all of our hardware elements. The programming environment used for this project is Microsoft's Visual Studio IDE in conjunction with the Visual Micro extension that uses Arduino's standard IDE in the background.

The software for our device is developed using miscellaneous default Arduino libraries such as SPI.h for facilitating communication via the SPI protocol and Wire.h for facilitating communication via the I2C protocol. The other two sets of libraries are specific Espressif as well as Adafruit libraries.

The first Espressif library used in this project is the ESP8266WiFi.h library based on Arduino's WiFi library used here for facilitating the connection between individual nodes. The second Espressif library is the espnow.h library which enables peer-to-peer communication without the need to connect to a server or the Internet. Using these two libraries our devices are capable of communicating with each other via previously known and software established MAC addresses through which they are capable of sending packets of information to specific destinations between nodes without the need for handshakes or packet receive acknowledgements.

Broadcast MAC address setting code in Arduino C:

uint8_t broadcastAddress[] = {0xA4, 0xE5, 0x7C, 0xB5, 0xDF, 0x9C};

This project also uses five Adafruit libraries developed by the open-source hardware company Adafruit, a popular hobby electronics manufacturer that offers free software and documentation for many Arduino and other microcontroller compatible hardware components. To begin with, our Nokia 5110 display is controlled using the Adafruit_GFX.h and Adafruit_PCD8544.h libraries which allow for drawing text, shapes and images as well as reliable refreshing of the LCD through its onboard controller IC. These libraries also come with premade dot matrix fonts while offering support for using custom libraries such as our custom Cyrillic font for displaying text in the Macedonian language. The Adafruit_Sensor.h and Adafruit_HMC5882_U.h libraries facilitate the calibration and readout of the GY-271 magnetometer, and lastly the Adafruit_PCF8575.h allows for reading the input pins of our PCF8575 hardware expander.

4 FUNCTIONALITY, DEMONSTRATION AND RESULTS

For this project we have developed a custom printed circuit board (Figure 4) using the Eagle CAD electronic design automation (EDA) software in order to deliver a standalone, compact and aesthetic device that can be soldered together and assembled creating a single node that can connect with other nodes of the same kind and run complementing software for multiple users to play within the same competitive multiplayer game.

As a demo for this project we have decided to recreate one of the oldest retro arcade games – PONG. Pong is a two-player game with the screen set as a tennis or table tennis court where both players operate paddles on the opposite side of each other and are required to hit a moving ball to the other side of the screen avoiding it falling through their side of the screen. Traditionally the paddles in this game and its recreations are placed on the left and right sides, however due to the motion control functionality implemented in our device this demo has the players' paddles on the bottom and on the top of the screen for moving them by tilting the device left and right as opposed to moving the whole device up or down.

After both players have entered the game, only the host of the game is able to start it. When the game starts, both devices display the same image on the Nokia 5110 display. The code that does this perpetually draws the ball, paddles, court and score onto the screen using an X and a Y coordinate for each element of the game. While the court and score are stationary objects, the ball and paddles move while the value of the score updates for each missed ball hit.

The code between host and client varies in the source of these coordinates. For the host, the coordinates of its player come from either the GY-271 sensor or two of the push buttons (either left or right arrows or A and V buttons) while the coordinates for the other player come via information sent by the other device through the ESP-NOW protocol of its own GY-271 sensor or its own push buttons (again either left or right arrows or A and V buttons).

For the client both paddle coordinates are received from the host through the ESP-NOW protocol as well as the coordinates for the ball whose collision and direction are detected and operated only by the host device. The host device also updates the score and conveys this information along to the client.

This is done by sending a data structure filled with information such as coordinates, scores and flags which must be declared identically in both the host and client programs.

Example of the data structure in Arduino C:

```
typedef struct struct_message {
    byte ballX;
    byte ballY;
    byte paddle1X = 0;
    byte paddle2X = 0;
    byte control;
    byte score [2];
    byte checkStart;
} struct_message;
```

Throughout the code for both nodes, two data structures are constantly being updated, one being sent to while the other one being received from the other device. This is why for our project different nodes run different code and for functional operation one of the nodes must always be the host and others the clients essentially processing most of the gameplay mechanics only on one device and then streaming the same image on the second (or multiple) device.

Although it is possible to run the same code on both (or multiple) nodes at the same time, cross updating information between nodes and allowing them to individually detect collisions and calculate scores, latency between magnetometer or push-button readout cycles as well as LCD refresh rates can cause lag or disturb the synchronization between the ball and the paddles throughout both devices leading to mismatching object coordinates by one or

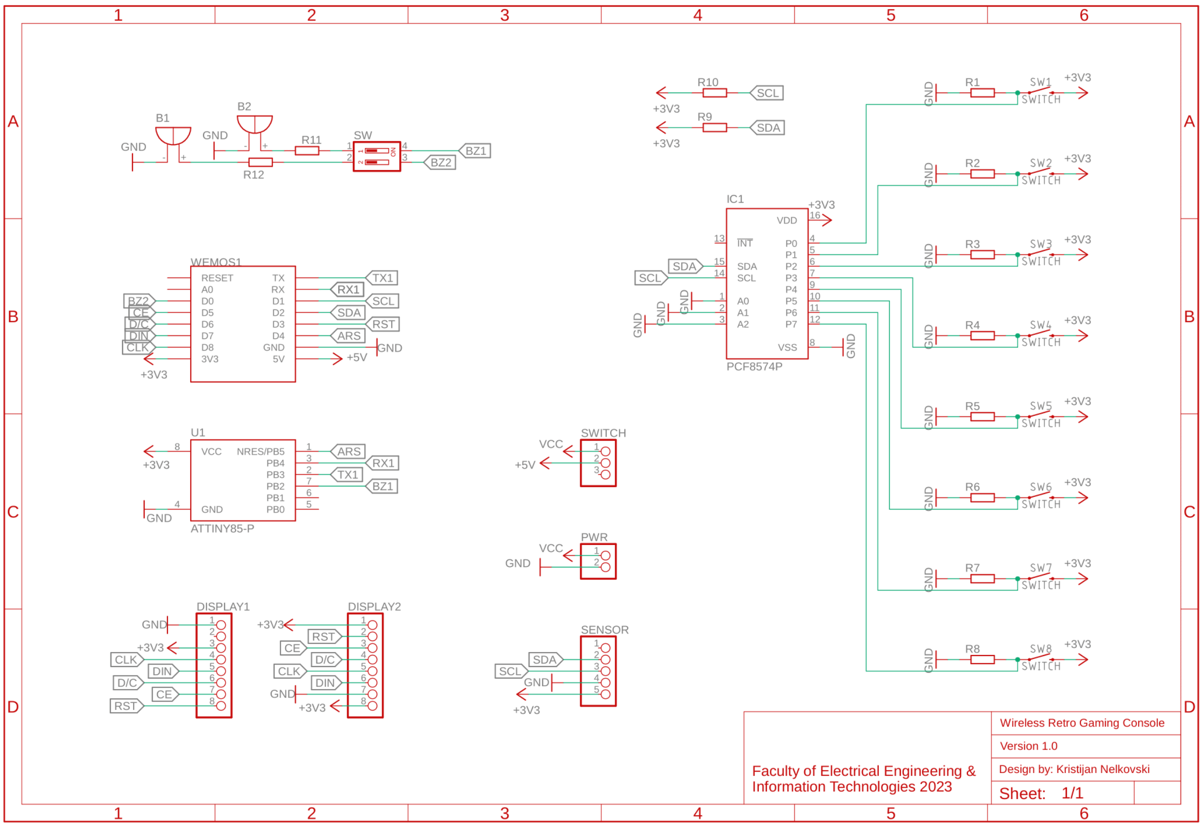
WEMOS1
U1
ATTINY85-P
IC1
PCF8574P
DISPLAY1
DISPLAY2
SWITCH
PWR
SENSOR
Wireless Retro Gaming Console
Version 1.0
Design by: Kristijan Nelkovski
Faculty of Electrical Engineering & Information Technologies 2023
Sheet: 1/1

Figure 4: The circuit board schematic designed in Eagle CAD.

more pixels which renders gameplay useless as even a single pixel carries a lot of weight on a small matrix of only 84 by 48 pixels. While there might still be latency and inconsistency between data transfer and writing images on the displays by using our method within the range of a couple dozen milliseconds, for the user this is far less noticeable and will never lead to mistakes of mismatched object coordinates, not even by a single pixel.

5 CONCLUSIONS

In this paper a handheld, networked and multiplayer video gaming device has been developed. With the use of ESP8266 based wireless microcontroller hardware and Arduino based software development tools, a viable recreation of the classic Pong arcade game was created as a two-player competitive video game demonstration with optional motion sensing controls, piezoelectric audio outputs as well as 1-bit (black and white) dot matrix LCD graphics.

For future development this system can be expanded with creating different types of video games for more than just two players and integrating higher resolution color LCD displays for more complex graphics, venturing closer into the territory of late 1990s handheld and mid 2000s Java mobile phone games as opposed to simpler 1970s and 1980s style arcade titles.

REFERENCES

[1] S. Staacks, “There oughta be a WiFi Game Boy cartridge,” December 2021, [Online]. Available: https://there.oughta.be/a/wifi-game-boy-cartridge.

[2] T. Roth, “Online Multiplayer on the Game Boy,” May 2021, github.com, youtube.com, [Online]. Available: https://github.com/stacksmashing, https://www.youtube.com/watch?v=KtHu693wE9o.

[3] M. Geddes, “Arduino Project Handbook,” vol. 2, No Starch Press, San Francisco, California, 2017.

[4] H. Wiguna, “How to write a game,” June 2016, instructables.com, youtube.com, [Online]. Available: https://www.instructables.com/HariFun-136-How-to-Write-a-Game/, https://www.youtube.com/watch?v=Xzx-cp60T7k.

[5] K.J.M. Kiili., K. Devlin, and J. Multisilta, “Editorial: is Game-Based Math Learning Finally Coming of Age?”, int.l Journal of Serious Games, vol. 2, nr. 4., 2015, doi: 10.17083/IJSG.V2I4.109.

[6] Y.A. Badamasi, “The working principle of an Arduino”, 11th Int. Conf. on Electronics, Computer and Computation (ICECCO), October 2014, doi: 10.1109/ICECCO.2014.6997578.

[7] D. Abbott, “Game-based learning for postgraduates: an empirical study of an educational game to teach research skills.”, High Educ Pedagog. 2019, doi: 10.1080/23752696.2019.1629825.

[8] J. Hartley, “Toshiba progress towards sensory control in real time,” Indust. Robot, vol. 14, no 1, pp. 50–52, 1987.

[9] R.L. Anderson, “A Robot Ping-Pong Player: Experiments in Real Time Control.”, Cambridge, MA: MIT Press, 1987.

[10] E.R. Davies, “Machine Vision.” New York: Academic, 1997.

[11] A. Rosenfeld and A. Kak, “Digital Picture Processing”, vol. 1–2. New York: Academic Press, 1982.