



SCARAB: Research Document

For submission to SETU

Author: Stuart Rossiter

Student Number: C00284845

Course: BSc (Hons.) Software Development

Supervisor: Joseph Kehoe

Contents

1. Abstract	4
2. Overview	5
3. SCARAB PC Program	6
3.1 Languages	6
3.1.1 C++	6
3.1.2 C#	6
3.1.3 Python	7
4. SCARAB Device	8
4.1 Microcontrollers	8
4.1.1 Arduino	8
4.1.1.1 Arduino Mega 2560 Rev3	8
4.1.1.2 Arduino UNO R3	9
4.1.2 ESP32	9
4.1.2.1 ESP32-P4-Function-EV-Board	9
4.1.2.2 ESP32-C6-DevKitM-1	10
4.2 Languages	10
4.2.1 Arduino Language	10
4.2.2 MicroPython	11
4.2.3 TinyGo	11
4.3 Game Cartridges	12
4.3.1 Nintendo Entertainment System (NES)	12
4.3.2 Super Nintendo Entertainment System (SNES)	13
4.3.3 Nintendo 64 (N64)	14
4.3.4 Game Boy (GB)	15
4.3.5 Game Boy Color (GBC)	16
4.3.6 Game Boy Advance (GBA)	16
4.4 Electronic Components	17
4.4.1 Bus Transceiver	17
4.4.2 Buck Converter	17
4.4.3 USB-C Input Module	18
4.4.4 Cartridge Ports	18
4.4.5 Resistors	18
5. Similar Products	19
5.1 Open Source Cartridge Reader (OSCR)	19
5.2 Retrode2	19

6. Conclusion	20
7. Appendix.....	21
8. Glossary	22
9. Bibliography.....	23

1. Abstract

The SCARAB device and accompanying program are retro game preservation tools designed to check the health of retro game cartridges, and manage their save data. The device, built around a microcontroller, interfaces with the cartridges via cartridge port modules. These modules can be swapped in and out, allowing for a variety of cartridge types, and providing future expansion options. The SCARAB device can auto-detect the inserted modules, and can detect whether a cartridge is inserted. The SCARAB program provides a GUI for the user to interact with. It provides options to dump and restore save data, run a full diagnostic checkup on an inserted cartridge, and run individual tests. It also serves as a save browser, allowing the user to view the save files they have dumped from cartridges.

2. Overview

When it comes to researching for the SCARAB, the research needed to be split up into 2 parts. One part focuses on the GUI PC Program. This first part goes into the choice of language for developing the program, and discusses the libraries which have been considered for use with the language. The second part of the research focuses on the SCARAB Device itself. First and foremost, research has been done about microcontrollers, as the “brain” of the device. In addition, the architecture of game cartridges will be discussed, as they will need to be interfaced with by the microcontroller. Finally, various electronic components will be required, and they will be discussed accordingly.

3. SCARAB PC Program

3.1 Languages

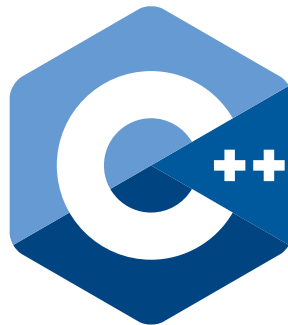
When it comes to choosing a language for the PC program portion of the SCARAB, a lot of thinking needed to be done. The list of requirements for a language are as follows:

- Does not need to be super performance oriented (e.g. Assembly).
- Needs to be able to interface with the SCARAB's microcontroller.
- Needs to have a good GUI library.
- Needs to be cross platform (Windows, Linux, MacOS).

Several languages are being considered for this.

3.1.1 C++

Developed by Bjarne Stroustrup, C++ is a cross-platform language with a high level of control over system resources and memory, due to it being an extension of the C language (W3Schools, 2011). It's an object-oriented language, facilitating structure and the reusability of code. C++ is known for being fast, as it is used in the likes of operating systems or embedded systems. While not necessary for the SCARAB, this is appreciated.



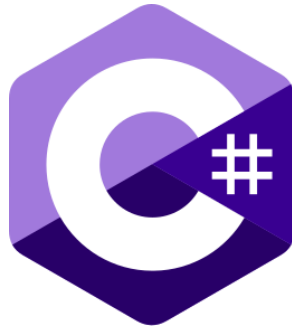
(Fig 3.1.1.1 – C++ Logo, (Wikimedia Commons, 2017))

With regards to microcontroller interfacing, C++ has a one-size-fits-all library: Boost.Asio. Boost.Asio is a part of the open-source Boost libraries, and it provides tools for asynchronous I/O, concurrency, and networking (Exito Technology, 2023). Boost.Asio provides means for communication over Wi-Fi and Serial connections, 2 prominent methods used by microcontrollers. Bluetooth can also be used, but it requires user implementation as it is not standard with Boost.Asio. The library also abstracts any platform-specific details, allowing for cross-platform support (Exito Technology, 2023).

Qt is a framework designed mainly around the creation of GUI based cross-platform applications. It contains a set of intuitive and modularized C++ library classes, and is loaded with APIs to simplify development (The Qt Company, n.d.). Tools such as Qt Design Studio, a UI design and development environment, also aid in development (The Qt Company, n.d.).

3.1.2 C#

Created by Microsoft, C# is a cross-platform general purpose language that runs on the .NET Framework (Microsoft, 2024). Much like C++, C# is in the C family of languages, leading to some familiar syntax such as ending lines with semi-colons, but with added features such as a “foreach” statement for any collection type.



(Fig 3.1.2.1 – C# Logo, (Microsoft, 2022))

C# has Wi-Fi and Serial connection modules built-in, allowing for easy communication with the SCARAB's microcontroller. With regards to GUI development, C# has an open-source cross-platform framework in Avalonia. Avalonia is a dotnet framework, considered by many to be the spiritual successor of Windows Presentation Foundation, an open-source UI framework for Windows (AvaloniaUI, 2024).

3.1.3 Python

Python, created by Guido van Rossum in 1991, is a general-purpose language with a simple syntax, similar to the English language (W3Schools, n.d.). Given its simplicity, it allows for rapid development, aided by its abundance of external libraries. It is a high-level interpreted language, and as such, speed is not a strong point. As mentioned above, however, speed is not paramount for the SCARAB, but it would be appreciated.



(Fig 3.1.3.1 – Python Logo, (Wikimedia Commons, 2008))

For microcontroller communication, Python can cover all bases. The 'socket' module allows for communication with Wi-Fi and Bluetooth enabled boards, opening sockets for low-level networking (Python, 2020). For Serial communications, the pySerial library is ideal. The pySerial module encapsulates serial port access, providing cross-platform backends for Python (pyserial, 2022).

For GUI development, Python has its own binding of the GUI toolkit Qt, developed by The Qt Company themselves. The binding, known as PySide, is cross-platform, much like Qt. The PySide binding is open source under the GPLv3 and Qt commercial license (pyside, 2021).

4. SCARAB Device

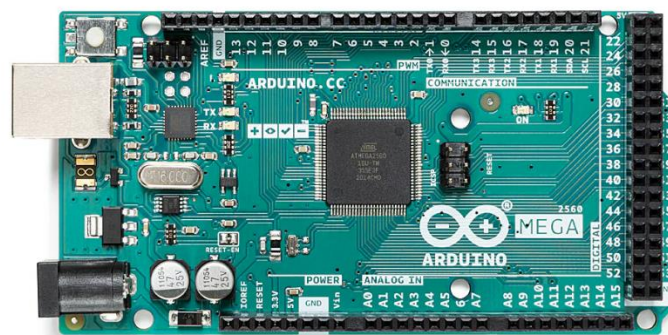
4.1 Microcontrollers

To serve as an interface between the cartridge port modules and the SCARAB PC program, some kind of microcontroller was deemed necessary. Microcontrollers are small computers on a single chip. They contain a processor core, RAM, and EEPROM, for storing programs to run. The purpose of a microcontroller is to manage a specific set of tasks within an embedded system, without the need for a complex operating system (Schneider & Smalley, 2024). This is perfect for the SCARAB, as there are only a handful of tasks necessary.

4.1.1 Arduino

Among the reasons for choosing Arduino, the main one is the variety of development boards. Several families of boards exist, such as Nano, MKR, UNO, Classic, and Mega, sporting over 30 different boards between them (Arduino, 2022). Given the sheer number of boards, and all their different configurations, there is sure to be a board suitable for the SCARAB. In addition to this, Arduino is inexpensive, has cross-platform support, and the software and hardware are completely open source. Choosing Arduino is not all that needs to be chosen, however. If Arduino is to be chosen, a specific board needs to be chosen. Below are some such boards.

4.1.1.1 Arduino Mega 2560 Rev3



(Fig 4.1.1.1.1 – Arduino Mega 2560, (Arduino, n.d.))

Based on the ATmega2560 microcontroller, the Arduino Mega 2560 Rev3 contains everything needed to support the ATmega2560, including 54 digital I/O pins, 16 analog inputs, 4 hardware serial ports, a 16MHz crystal oscillator, USB connection, and a power jack (Arduino, 2025). The Mega 2560 has enough GPIO pins to handle even the largest of cartridge ports, and the Arduino “port” system allows for the reading from/writing to 8 pins at a time, i.e. 1 byte, all in one single instruction. The Mega 2560 runs on 5V logic. Most cartridges run at 5V logic, with the occasional 3.3V cartridges, so bus transceivers would be needed in these scenarios.

4.1.1.2 Arduino UNO R3



(Fig 4.1.1.2.1 – Arduino UNO R3, (ARDUINO, 2021))

The Arduino UNO R3 is based around the ATmega328P, which is socketed on the board to allow for easy replacement of it in case of failure. The board supports the microcontroller with 14 digital I/O pins, 6 analog inputs, a 16MHz ceramic resonator, a USB connection, and a power jack (Arduino, 2022). To handle the cartridge I/O, the Arduino UNO would need some form of multiplexing and latches, to interface with different lines and to store those inputs/outputs. In addition to that, the UNO runs at 3.3V. The UNO R3 also runs on 5V logic, much like the Mega 2560. Most cartridges run at 5V logic, with the occasional 3.3V cartridges, so bus transceivers would be needed in these scenarios.

4.1.2 ESP32

The ESP32 family of chips and boards are developed by Espressif; a fabless silicon vendor based in China, and founded in 2008 (Cording, 2022). Centred around Wi-Fi and Bluetooth connectivity, the ESP32 chip is ideal for Internet of Things (IoT) devices, wearables, and smart home automation. Below are some ESP32 boards.

4.1.2.1 ESP32-P4-Function-EV-Board

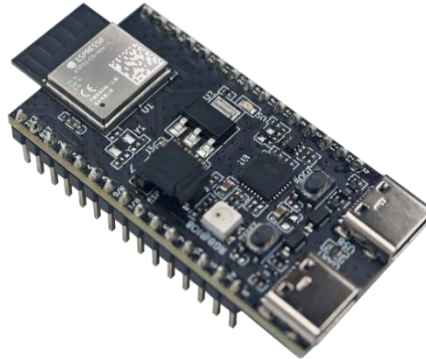


(Fig 4.1.2.1.1 – ESP32-P4-Function-EV-Board, (Espressif, 2016))

Clocking in at a maximum of 400MHz, the ESP32-P4 features a dual-core RISC-V processor, supporting the likes of USB 2.0, MIPI-CSI/DSI, H264 Encoder, and various other peripherals (Espressif, 2016). The board supports WiFi 6, Bluetooth 5.0, USB type A and C, Ethernet, and

has 55 GPIOs. It also runs at 3.3V, which will require boosting to 5V for some cartridge types. The ESP32-P4-Function-EV-Board seems to be overkill for the project, and it seems to be geared towards audio and video devices.

4.1.2.2 ESP32-C6-DevKitM-1



(Fig 4.1.2.1.1 – ESP32-C6-DevKitM-1, (Espressif, 2023))

The ESP32-C6-MINI-1(U) supports WiFi 6, Bluetooth 5.0, and maxes out at 160MHz (Espressif, 2023). With 22 GPIO pins, the board would need some form of multiplexing and latches, to interface with different lines and to store those inputs/outputs. Much like the ESP32-P4-Function-EV-Board, the ESP32-C6-DevKitM-1 is geared towards audio and video devices, or IoT usage.

4.2 Languages

There are a variety of languages supported by the above microcontrollers, all with their pros and cons:

4.2.1 Arduino Language



(Fig 4.2.1.1 – Arduino IDE Logo, (Wikimedia Commons, n.d.))

The Arduino programming language is essentially C++ with extra features. It serves as the default programming language for Arduino boards, for use with the Arduino IDE (Arduino, n.d.). As it is based on C++, the Arduino language is fast and lightweight. Support for ESP32 microcontrollers is available, using an Arduino core developed by Espressif, the creators of the ESP32 microcontrollers (Espressif, 2019). The Arduino language is also heavily documented, with code examples available directly from the Arduino IDEs (Arduino, 2025).

4.2.2 MicroPython



(Fig 4.2.2.1 – MicroPython Logo, (Wikimedia Commons, 2018))

MicroPython is an implementation of Python 3 that includes a small subset of the Python standard library, optimised to run on microcontrollers and in constrained environments (MicroPython, 2024). Despite all these optimizations, it still has some drawbacks. The main drawback is its compatibility. Among the Arduino boards, only a select few are even compatible (Arduino, 2025). ESP32 boards surpass the specifications for MicroPython, however, and it would be suited to them.

4.2.3 TinyGo



(Fig 4.2.3.1 – TinyGo Logo, (TinyGo, 2024))

Bringing concurrency to embedded systems, TinyGo is a port of the Go language to microcontrollers. Unfortunately, it has several flaws as of current. Many microcontrollers are not well-supported, such as the ATmega328P seen in the Arduino UNO, or the ESP32 seen in the ESP32-P4-Function-EV-Board (TinyGo, 2024). It lacks Wi-Fi and Bluetooth support for the ESP32 – one of the microcontroller’s main selling points. In addition to this, the main selling point of Go is its concurrency, but this is irrelevant for microcontrollers possessing only a single core (most Arduino boards). For this reason, there is no reason to use TinyGo on single-core microcontrollers, as it can not be leveraged to its full potential.

4.3 Game Cartridges

The SCARAB is intended to interface with many different cartridges through its module boards. The architecture of these cartridges varies, so measures must be taken on a system-by-system basis.

4.3.1 Nintendo Entertainment System (NES)

Released in 1983 in Japan as the Famicom, and later in Europe in 1986 (Nintendo, 2016), the NES single-handedly saved the gaming industry. The cartridges, known as Game Paks (a recurring name for Nintendo), have since become iconic.



(Fig. 3.3.1.1 NES Cartridge External View)



(Fig. 3.3.1.2 NES Cartridge Internal View)

Earlier games, such as Super Mario Bros, were simple to read and write to, as they contained 2 important chips: one containing 32KB of Program ROM, and the other containing 8KB of Character ROM. These were part of a board revision known as NROM-256 (NESdev Wiki, 2024). The problem with reading later cartridges comes from the introduction of “Mapper Chips”. These chips allowed the NES to bypass its 16-bit address bus limit, by swapping the currently accessible ROM data on the cartridge, allowing for much bigger games (NESdev Wiki, 2023). All but 20 games are covered by the Action 53, MMC1, and MMC3 families, so implementing the 256 registered mappers is not a goal of this project.

(Front/top)	NES	Cart	(back/bottom)	NES
+5V	--	[36 72]	--	GND
CIC toMB	<	[35 71]	<	CIC CLK
CIC toPak	->	[34 70]	<	CIC +RST
PPU D3	<	[33 69]	<	PPU D4
PPU D2	<	[32 68]	<	PPU D5
PPU D1	<	[31 67]	<	PPU D6
PPU D0	<	[30 66]	<	PPU D7
PPU A0	->	[29 65]	<	PPU A13
PPU A1	->	[28 64]	<	PPU A12
PPU A2	->	[27 63]	<	PPU A10
PPU A3	->	[26 62]	<	PPU A11
PPU A4	->	[25 61]	<	PPU A9
PPU A5	->	[24 60]	<	PPU A8
PPU A6	->	[23 59]	<	PPU A7
CIRAM A10	<	[22 58]	<	PPU /A13
PPU /RD	->	[21 57]	->	CIRAM /CE
EXP 4	[20 56]	<	PPU /vR	
EXP 3	[19 55]		EXP 5	
EXP 2	[18 54]		EXP 6	
EXP 1	[17 53]		EXP 7	
EXP 0	[16 52]		EXP 8	
/IRQ	<	[15 51]		EXP 9
CPU R/W	->	[14 50]	<	/ROMSEL (/A15 + /M2)
CPU A0	->	[13 49]	<	CPU D0
CPU A1	->	[12 48]	<	CPU D1
CPU A2	->	[11 47]	<	CPU D2
CPU A3	->	[10 46]	<	CPU D3
CPU A4	->	[09 45]	<	CPU D4
CPU A5	->	[08 44]	<	CPU D5
CPU A6	->	[07 43]	<	CPU D6
CPU A7	->	[06 42]	<	CPU D7
CPU A8	->	[05 41]	<	CPU A14
CPU A9	->	[04 40]	<	CPU A13
CPU A10	->	[03 39]	<	CPU A12
CPU A11	->	[02 38]	<	M2
GND	--	[01 37]	<	SYSTEM CLK

Figure 3.3.1.3 shows the pinout of the NES cartridge connector. Most of the important pins for this project come in the format “XYZ”, where XXX is either CPU (leading to Program ROM), or PPU (leading to Character ROM), Y is either A for address line or D for data line, and Z is the number for the line. Other important pins include the 5V and GND lines, M2 for the mapper chips, /ROMSEL, CPU R/W, PPU /RD, and CIC +RST (Reset).

(Fig 3.3.1.3 NES cartridge pinout (NESdev Wiki, 2023))

4.3.2 Super Nintendo Entertainment System (SNES)

In 1990, Nintendo Japan released the Super Famicom, the 16-bit successor to the Famicom. This was later released in Europe in June 1992 as the SNES (Nintendo, 2016).



(Fig. 3.3.2.1 SNES Cartridges External View)



(Fig. 3.3.2.2 SNES Cartridges Internal View)

Many games are straightforward to read from and interface with. The problems arise with some of the “Enhancement Chips” that were seen in later games. Some enhancement chips hold no effect on the reading of cartridges, such as the Super FX chip (seen in Starwing, see MARIO CHIP-1 in Fig 3.3.2.2). The Super Accelerator 1 (SA1) chip, however, is troublesome.

The SA1 is a coprocessor, running at 10.74 MHz: 4 times faster than the speed of the SNES CPU (SnesLab, 2025). As seen in figure 3.3.2.3, it has pins which interface directly with ROM, and has pins for CIC data. This is where the problem lies. Some ROM data is locked behind the SA1 coprocessor, as it contains a Super MMC memory mapper chip. The SA1 begins in a sleeping state, to be woken by the SNES CPU, which will not happen if the CIC lockout chip and correct clock timing are not made available to the SA1. This can be overcome, as seen in how the Sanni Open-Source Cartridge Reader works (sanni, 2024). For the sake of this project, the SA1 bypass will be skipped. It may be implemented down the line.

```

/IRQ <- / 128 \ <- RD
D7 <- / 2 127 \ <- Region
D3 <- / 3 126 \ <- /NR
D6 <- / 4 125 \ <- CIC Data 1
D2 <- / 5 124 \ <- CIC Data 2
D5 <- / 6 123 \ <- CIC Reset
D1 <- / 7 122 \ <- CIC Clock
D4 <- / 8 121 \ <- SYSCCLK
D0 <- / 9 120 \ <- /RESET
VCC -- / 10 119 -- VCC
GND -- / 11 118 -- GND
A23 -> / 12 117 \ <- SRAM D7
A8 -> / 13 116 \ <- SRAM D6
A22 -> / 14 115 \ <- SRAM D5
A1 -> / 15 114 \ <- SRAM D4
A21 -> / 16 113 \ <- SRAM D3
A2 -> / 17 112 \ <- SRAM D2
A20 -> / 18 111 \ <- SRAM D1
A3 -> / 19 110 \ <- SRAM D0
A19 -> / 20 109 \ <- SRAM /NR
A4 -> / 21 108 \ <- SRAM /RD
A18 -> / 22 107 \ <- SRAM A19?
A5 -> / 23 106 \ <- SRAM A17?
A17 -> / 24 105 \ <- SRAM A15
A6 -> / 25 104 \ <- SRAM A18?
A16 -> / 26 103 \ <- SRAM A13
A7 -> / 27 102 / <- SRAM A8
A15 -> / 28 101 / <- VCC
A8 -> / 29 100 / <- GND
A14 -> / 30 99 / <- SRAM A9
A9 -> / 31 98 / <- SRAM A11
A13 -> / 32 97 / <- SRAM A10
A10 -> / 33 96 / <- SRAM A0
A12 -> / 34 95 / <- SRAM A1
A11 -> / 35 94 / <- SRAM A2
VCC -- / 36 93 / <- SRAM A3
GND -- / 37 92 / <- SRAM A4
REFRESH -> / 38 91 / <- SRAM A5
GND -- / 39 90 / <- SRAM A6
CLK -> \ 40 89 / <- SRAM A7
CLK -> \ 41 88 / <- SRAM A12
GND -- \ 42 87 / <- SRAM A14
ROM D15 -> \ 43 86 / <- SRAM A16?
ROM D7 -> \ 44 85 / <- GND?
ROM D14 -> \ 45 84 / <- GND
ROM D6 -> \ 46 83 / <- VCC
ROM D11 -> \ 47 82 / <- GND?
ROM D3 -> \ 48 81 / <- ROM A23?
ROM D10 -> \ 49 80 / <- ROM A22
ROM D2 -> \ 50 79 / <- ROM A21
ROM D13 -> \ 51 78 / <- ROM A20
ROM D5 -> \ 52 77 / <- ROM A18
ROM D12 -> \ 53 76 / <- ROM A19
ROM D4 -> \ 54 75 / <- ROM A17
ROM D9 -> \ 55 74 / <- ROM A16
ROM D1 -> \ 56 73 / <- ROM A15
ROM D8 -> \ 57 72 / <- ROM A14
ROM D0 -> \ 58 71 / <- ROM A13
ROM A1 <- \ 59 70 / <- ROM A12
ROM A2 <- \ 60 69 / <- ROM A11
ROM A3 <- \ 61 68 / <- ROM A10
ROM A4 <- \ 62 67 / <- ROM A9
ROM A5 <- \ 63 66 / <- ROM A8
ROM A6 <- \ 64 65 / <- ROM A7

```

(Fig. 3.3.2.3 SNES SA1 Chip Pinout (SNESdev Wiki, 2025))

SNES (front)	Cart	SNES (back)
SYSTEM CLK ->	01 32	<- /NRAMSEL
EXPAND <->	02 33	<- REFRESH
PA6 ->	03 34	<- PA7
/PRD ->	04 35	<- /PWR
GND --	05 36	-- GND
CPU A11 ->	06 37	<- CPU A12
CPU A10 ->	07 38	<- CPU A13
CPU A9 ->	08 39	<- CPU A14
CPU A8 ->	09 40	<- CPU A15
CPU A7 ->	10 41	<- CPU A16
CPU A6 ->	11 42	<- CPU A17
CPU A5 ->	12 43	<- CPU A18
CPU A4 ->	13 44	<- CPU A19
CPU A3 ->	14 45	<- CPU A20
CPU A2 ->	15 46	<- CPU A21
CPU A1 ->	16 47	<- CPU A22
CPU A0 ->	17 48	<- CPU A23
/IRQ <-	18 49	<- /ROMSEL
CPU D0 <->	19 50	<-> CPU D4
CPU D1 <->	20 51	<-> CPU D5
CPU D2 <->	21 52	<-> CPU D6
CPU D3 <->	22 53	<-> CPU D7
CPU /RD ->	23 54	<- CPU /WR
CIC data 1 <->	24 55	<-> CIC data 2
/RESET <->	26 57	<- PHI2
+5V --	27 58	-- +5V
PA0 ->	28 59	<- PA1
PA2 ->	29 60	<- PA3
PA4 ->	30 61	<- PA5
left audio in <-	31 62	-> right audio in

(Fig 3.3.2.4 SNES Connector Pinout (SNESdev Wiki, 2022))

4.3.3 Nintendo 64 (N64)

1996 graced Japanese store shelves with the world's first 64-bit home videogame system: the Nintendo 64. The following year, it was released in Europe, to critical acclaim (Nintendo, 2016).



(Fig. 3.3.3.1 N64 Cartridge External View)



(Fig. 3.3.3.2 N64 Cartridge Internal View)

The N64 marks the first cartridge type in my research to use 3.3V power and logic, as opposed to 5V. This is where bus transceivers and a buck converter will be needed see (3.4.1 and 1.4.2).

Cartridge

Looking at the top of the console with controller ports facing you. Pin 1 is lower left, pin 26 upper left.

Pin	Name	Description	Pin	Name	Description
1	GND		26	GND	
2	GND		27	GND	
3	AD15		28	AD0	
4	AD14		29	AD1	
5	AD13		30	AD2	
6	GND		31	GND	
7	AD12		32	AD3	
8	WR	Write	33	ALE_L	Address latch low bits
9	3.3v		34	3.3v	
10	RD	Read	35	ALE_H	Address latch high bits
11	AD11		36	AD4	
12	AD10		37	AD5	
13	12v		38	12v	
14	12v	expansion port	39	12v	Expansion port
15	AD9		40	AD6	
16	AD8		41	AD7	
17	3.3v		42	3.3v	
18	CIC_15	CIC → PIF Data	43	CIC_14	PIF ← CIC Data
19	CIC_11	1.95MHz clock (PIF driven)	44	?	JTAG_CLK_R4300 or INT
20	RESET		45	NMI	
21	EEPROM_DAT		46	CSYNC	
22	GND		47	GND	
23	GND		48	GND	
24	SL	Audio Left	49	SR	Audio Right
25	GND		50	GND	

Figure 3.3.3.3 shows the pinout of the N64 cartridge. AD0-15 are multifunctional. They control the upper 2 bytes of the address input, the lower 2 bytes of the address input, and the 2 bytes of data output. The addresses are latched within the cartridge by the ALE_L and ALE_H pins, and the data is read with the RD pin.

In addition to those, the GND, 3.3V, RESET, and EEPROM_DAT lines will likely be required.

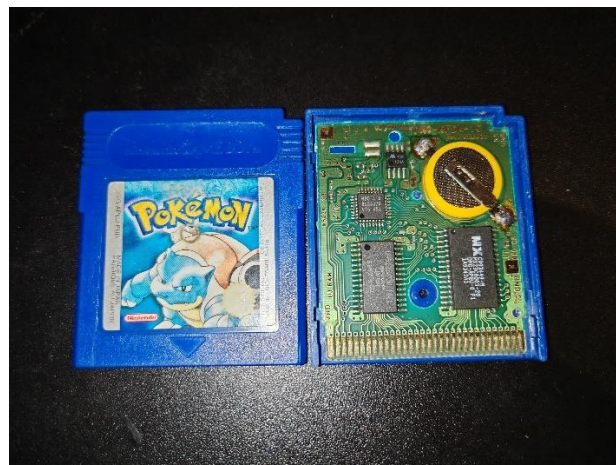
(Fig. 3.3.3.3 N64 Connector Pinout (ConsoleMods Wiki, 2022))

4.3.4 Game Boy (GB)

In 1989, Nintendo Japan released the first handheld game system with interchangeable game cartridges. Its name? The Game Boy. It released alongside Tetris, one of its bestselling games. In 1990, the Game Boy came to Europe, and with it, Nintendo of Europe was formed (Nintendo, 2016).



(Fig. 3.3.4.1 Game Boy Cartridge External View)



(Fig. 3.3.4.2 Game Boy Cartridge Internal View)

The Game Boy, much like the NES, had mapper chips of its own. Games larger than 32KB needed to use these mapper chips. These chips allowed for “banks” of ROM data to be switched in and out, for access via the address lines. This could be achieved by writing specific values to areas on the cartridge (Retrocomputing Stack Exchange, 2019). Apart from that, addressing the ROM is rather straightforward.

Cartridge

Pins numbered left to right, cartridge label up.

Pin	Function	Pin	Function	Pin	Function	Pin	Function
01	+5V	11	Address Bit 5	21	Address Bit 15	31	AUDIO IN
02	CLOCK	12	Address Bit 6	22	Data Bit 0	32	GROUND
03	WRITE	13	Address Bit 7	23	Data Bit 1		
04	READ	14	Address Bit 8	24	Data Bit 2		
05	CHIP SELECT	15	Address Bit 9	25	Data Bit 3		
06	Address Bit 0	16	Address Bit 10	26	Data Bit 4		
07	Address Bit 1	17	Address Bit 11	27	Data Bit 5		
08	Address Bit 2	18	Address Bit 12	28	Data Bit 6		
09	Address Bit 3	19	Address Bit 13	29	Data Bit 7		
10	Address Bit 4	20	Address Bit 14	30	RESET		

(Fig. 3.3.4.3 GB Connector Pinout (ConsoleMods Wiki, 2022))

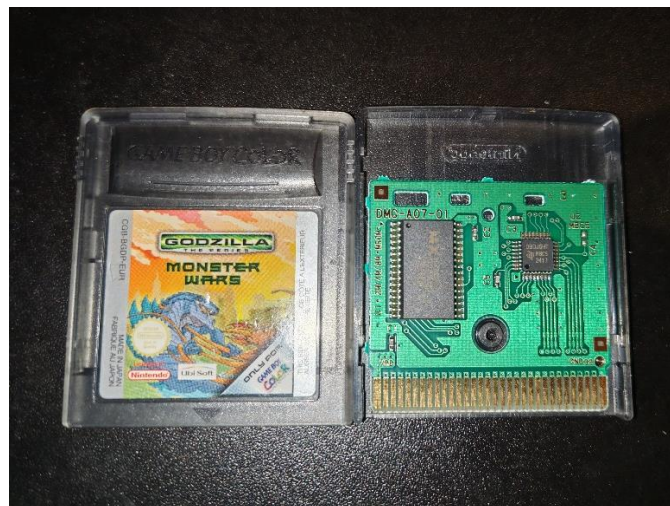
Above is the pinout for the Game Boy cartridge. All but AUDIO IN will likely be required.

4.3.5 Game Boy Color (GBC)

1998 saw the release of the next step up from the Game Boy, in the Game Boy Color (Nintendo, 2016).



(Fig. 3.3.5.1 GBC Cartridge External View)

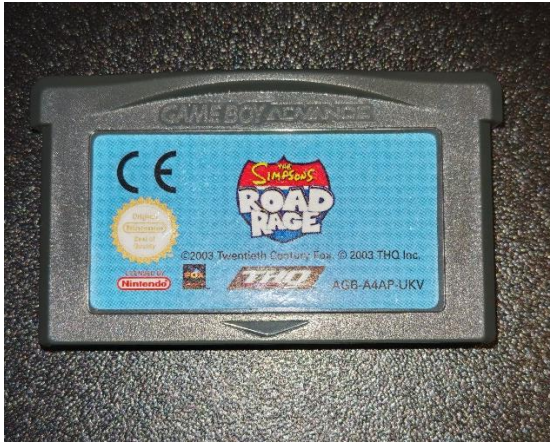


(Fig. 3.3.5.2 GBC Cartridge Internal View)

From my research, the Game Boy Color cartridges utilise the same addressing and banking systems that the original Game Boy used. This makes things easier, as they are essentially Game Boy games, but bigger.

4.3.6 Game Boy Advance (GBA)

The Game Boy Advance released worldwide in 2001, being marked as the fastest selling console ever with over 500k units sold in its first week in Europe. 2 years later, we would see it take on a new clam-shell form; the Game Boy Advance SP (Nintendo, 2016).



(Fig. 3.3.6.1 GBA Cartridge External View)



(Fig. 3.3.6.2 GBA Cartridge Internal View)

GBA cartridges are the second cartridges to use 3.3V power and logic. So long as the first read is non-sequential, it's possible to sequentially read from the ROM by pulsing the "RD" pin repeatedly, to a point (Ziegler, 2025).

Pin	Function	Pin	Function	Pin	Function	Pin	Function
01	+3.3V	11	Data/Address Bit 5	21	Data/Address Bit 15	31	IRQ
02	CLOCK	12	Data/Address Bit 6	22	Address Bit 16	32	GROUND
03	WRITE	13	Data/Address Bit 7	23	Address Bit 17		
04	READ	14	Data/Address Bit 8	24	Address Bit 18		
05	CHIP SELECT	15	Data/Address Bit 9	25	Address Bit 19		
06	Data/Address Bit 0	16	Data/Address Bit 10	26	Address Bit 20		
07	Data/Address Bit 1	17	Data/Address Bit 11	27	Address Bit 21		
08	Data/Address Bit 2	18	Data/Address Bit 12	28	Address Bit 22		
09	Data/Address Bit 3	19	Data/Address Bit 13	29	Address Bit 23		
10	Data/Address Bit 4	20	Data/Address Bit 14	30	CHIP SELECT 2		

(Fig. 3.3. GBA Connector Pinout (ConsoleMods Wiki, 2022))

Reading the ROM is simple enough. The save data depends on the format, however. If it is battery backed SRAM, it's simple reads and writes. Flash storage requires specific "commands" sent as writes to memory (Korth, 2024). EEPROM is easily the most annoying of the 3. To make a long story short, it handles data serially, i.e. one bit at a time (DenSinH, 2021).

4.4 Electronic Components

4.4.1 Bus Transceiver

As the GBA and N64 cartridges utilise 3.3V logic, as opposed to the Arduino's 5V logic, some form of voltage lowering was required. The method of choice was the SN74LVC245AN Octal Bus Transceiver. The bus transceiver allows for bi-directional voltage modulation, converting 5V to 3.3V on one side, and 3.3V to 5V on the other. This feature lets the device function as a translator in mixed 3.3V and 5V environments (Texas Instruments, 2015). These transceivers will allow the Arduino's 5V logic to interface with the 3.3V logic of the GBA and N64 cartridges.

4.4.2 Buck Converter

Not only do the GBA and N64 cartridges use 3.3V logic, but they utilise 3.3V VCC too. The transceivers would not work for the level of amps required, so a new solution was needed. This

is where buck converters are introduced. Buck converters convert a DC voltage to a lower DC voltage, such as 5V to 3.3V in this case (Yates, 2024).

4.4.3 USB-C Input Module

While a microcontroller has enough power to interface with the cartridge, it doesn't support the Amps necessary to also power it. Using a USB-C cable, the SCARAB could draw the power necessary from a USB port on the PC. While this will require 2 USB cables to be connected between the SCARAB and the PC, it's preferable to the alternative of batteries.

4.4.4 Cartridge Ports

A microcontroller may have many I/O pins, but cartridges can't be inserted directly into pins. Cartridge ports are widely available, mostly known by their pin count. For example, the NES port is known as a 72-pin connector. These ports have pins which can be slotted into pin connectors, or directly into PCBs, which will be the case for the cartridge port modules.

4.4.5 Resistors

Resistors are passive devices used to control the flow of current in a circuit. Controlling current flow helps to prevent damage to fragile components.

4.4.6 EEPROM

EEPROM chips will be used for identification of modules. They can work using serial connections, which the microcontrollers can either support natively or spoof with the correct timing.

5. Similar Products

5.1 Open Source Cartridge Reader (OSCR)

Created by Sanni, the Open Source Cartridge Reader is a device designed to allow the dumping of cartridge ROMs and Save Data, along with the restoring of saves.

Similarities:

- Both the SCARAB and OSCR will dump and restore save data.
- Both the SCARAB and OSCR use microcontrollers.
- Both the SCARAB and OSCR can calculate the checksum of a cartridge.

Differences:

- The OSCR does not check the checksum against a known valid checksum.
- The OSCR does not interface with a PC. It's a standalone device with an LCD and SD card. The SCARAB has a PC program it interacts with.
- The SCARAB has automatic voltage detection as standard, with the OSCR it is a modification.
- The OSCR focuses on software preservation, and the SCARAB focuses more on hardware preservation.
- The OSCR solders in multiple cartridge ports to the device itself, where the SCARAB uses a plug-and-play module system.

5.2 Retrode2

The Retrode2 is a USB interface for retro cartridges and controllers. It allows users to play their favourite retro games on PC emulators.

Similarities:

- The Retrode2 allows for save dumping and restoring.
- The Retrode2 interfaces with a PC.

Differences:

- The Retrode2 does not check the integrity of the cartridges.
- The Retrode2 serves as an adaptor for SNES and Sega controllers to PC.
- The Retrode2's main purpose is to bridge the gap between the cartridges and emulators on modern machines, where the SCARAB's main purpose is hardware preservation.

6. Conclusion

Throughout this document, many aspects of the SCARAB have been researched and discussed. First was the GUI Program, where the candidates for language were discussed. Python has been chosen as the language for the program, with the pyserial and PySide libraries. As far as the SCARAB device is concerned, all aspects of it have been researched. The microcontroller chosen is the Arduino Mega 2560. The language of choice for the Arduino is the default C++ based Arduino language. Space conservation is important, so MicroPython won't do, and TinyGo is focused on concurrency, but the Mega 2560 only has one core. Cartridge pinouts and architecture were also discussed. This was a very important topic, as the Arduino must interface with the cartridges. Knowing which pins are necessary, and where they lie on the connectors, along with any other oddities relating to the cartridges is paramount. Extra components are required, as research had found. These were simple things, such as bus transceivers and buck converters to handle voltage differences, cartridge ports to provide a more familiar interface for the Arduino, USB-C inputs for power, and resistors for many different things.

This document has established several ideas which are important to remember for the development of this project:

- Cartridges are complex. Voltages are different. Pins are different. Pin count is different. Some have extra chips such as mapper chips, EEPROMs, and even CPUs.
- Sometimes, simplicity is ok. For example, C++ may have better speed and memory control than Python, but none of that is needed for this project.

7. Appendix

8. Glossary

SCARAB – Save and Cartridge Aid Requiring Adapter Boards

UI – User Interface

GUI – Graphical User Interface

RAM – Random Access Memory

EEPROM – Electrically Erasable Programmable Read-Only Memory

I/O – Input/Output

ROM – Read Only Memory

CPU – Central Processing Unit

PPU – Pixel Processing Unit

SRAM – Static Random Access Memory

OSCR – Open Source Cartridge Reader

LCD – Liquid Crystal Display

SD – Secure Digital

USB – Universal Serial Bus

DC – Direct Current

CIC – Checking Integrated Circuit

IoT – Internet of Things

GPIO – General Purpose Input Output

9. Bibliography

Anon., 2025. *PySide*. [Online]

Available at: <https://www.pythonguis.com/topics/pyside>

[Accessed 12 October 2025].

ARDUINO, 2021. *Arduino Uno Rev3*. [Online]

Available at: <https://store.arduino.cc/products/arduino-uno-rev3>

[Accessed 9 November 2025].

Arduino, 2022. *Arduino Hardware*. [Online]

Available at: <https://www.arduino.cc/en/hardware>

[Accessed 9 October 2025].

Arduino, 2022. *UNO R3 | Arduino Documentation*. [Online]

Available at: <https://docs.arduino.cc/hardware/uno-rev3>

[Accessed 9 November 2025].

Arduino, 2025. *Arduino® MEGA 2560 Rev3*. [Online]

Available at: <https://docs.arduino.cc/resources/datasheets/A000067-datasheet.pdf>

[Accessed 10 October 2025].

Arduino, 2025. *Built-in Examples*. [Online]

Available at: <https://docs.arduino.cc/built-in-examples>

[Accessed 9 November 2025].

Arduino, 2025. *MicroPython*. [Online]

Available at: <https://docs.arduino.cc/micropython>

[Accessed 9 November 2025].

Arduino, n.d. *Arduino IDE*. [Online]

Available at: <https://docs.arduino.cc/software/ide>

[Accessed 9 November 2025].

Arduino, n.d. *Arduino Mega 2560*. [Online]

Available at: [https://store-](https://store-usa.arduino.cc/cdn/shop/files/A000067_00.front_1000x750.jpg?v=1727102662)

[usa.arduino.cc/cdn/shop/files/A000067_00.front_1000x750.jpg?v=1727102662](https://store-usa.arduino.cc/cdn/shop/files/A000067_00.front_1000x750.jpg?v=1727102662)

[Accessed 11 October 2025].

AvaloniaUI, 2024. *GitHub - AvaloniaUI/Avalonia: Develop Desktop, Embedded, Mobile and WebAssembly apps with C# and XAML. The most popular .NET UI client technology*. [Online]

Available at: <https://github.com/avaloniaui/avalonia>

[Accessed 9 November 2025].

ConsoleMods Wiki, 2022. *Game Boy Connector Pinouts*. [Online]

Available at: https://consolemods.org/wiki/Game_Boy:Connector_Pinouts

[Accessed 12 October 2025].

ConsoleMods Wiki, 2022. *GBA Connector Pinouts*. [Online]

Available at: https://consolemods.org/wiki/GBA:Connector_Pinouts

[Accessed 12 October 2025].

ConsoleMods Wiki, 2022. *N64 Connector Pinouts*. [Online]
Available at: https://consolemods.org/wiki/N64:Connector_Pinouts
[Accessed 12 October 2025].

Cording, S., 2022. *What Is the ESP32? Its Brief History and How to Get Started*. [Online]
Available at: <https://www.elektormagazine.com/articles/what-is-the-esp32>
[Accessed 9 November 2025].

Dahl, Ø. N., 2023. *What Is A Resistor And What Does It Do?*. [Online]
Available at: <https://www.build-electronic-circuits.com/what-is-a-resistor>
[Accessed 11 October 2025].

DenSinH, 2021. *[GBA] EEPROM Save Type*. [Online]
Available at: <https://densinh.github.io/DenSinH/emulation/2021/02/01/gba-EEPROM.html>
[Accessed 12 October 2025].

Espressif, 2016. *ESP32-P4-Function-EV-Board v1.5.2 - ESP32-P4 - — esp-dev-kits latest documentation*. [Online]
Available at: https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32p4/esp32-p4-function-ev-board/user_guide.html
[Accessed 9 November 2025].

Espressif, 2019. *espressif/arduino-esp32*. [Online]
Available at: <https://github.com/espressif/arduino-esp32>
[Accessed 9 November 2025].

Espressif, 2023. *ESP32-C6-DevKitM-1 - ESP32-C6 - — esp-dev-kits latest documentation*. [Online]
Available at: https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32c6/esp32-c6-devkitm-1/user_guide.html
[Accessed 9 November 2025].

Exito Technology, 2023. *Boost.Asio: A Powerful Networking Library for C++*. [Online]
Available at: <https://medium.com/@extio/boost-asio-a-powerful-networking-library-for-c-e86367c2568d>
[Accessed 9 November 2025].

GitHub, 2022. *pyserial*. [Online]
Available at: <https://github.com/pyserial/pyserial>
[Accessed 12 October 2025].

Korth, M., 2024. *GBATEK - GBA/NDS Technical Info*. [Online]
Available at: <https://problemkaputt.de/gbatek.htm#gbacartbackupflashrom>
[Accessed 12 October 2025].

Kumar, A., 2023. *Why do We Use the Arduino Programming Language? How is it Helpful?*. [Online]
Available at: <https://emeritus.org/blog/coding-arduino-programming-language>
[Accessed 12 October 2025].

MicroPython, 2024. *The MicroPython project*. [Online]
Available at: <https://github.com/micropython/micropython>
[Accessed 9 November 2025].

Microsoft, 2022. *File:Logo C sharp.svg*. [Online]
Available at: https://commons.wikimedia.org/wiki/File:Logo_C_sharp.svg
[Accessed 09 November 2025].

Microsoft, 2024. *Overview - A tour of C#*. [Online]
Available at: <https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/overview>
[Accessed 9 November 2025].

NESdev Wiki, 2023. *Cartridge connector*. [Online]
Available at: https://www.nesdev.org/wiki/Cartridge_connector
[Accessed 11 October 2025].

NESdev Wiki, 2023. *Mapper*. [Online]
Available at: <https://www.nesdev.org/wiki/Mapper>
[Accessed 11 October 2025].

NESdev Wiki, 2024. *NROM*. [Online]
Available at: <https://www.nesdev.org/wiki/NROM>
[Accessed 11 October 2025].

Nintendo, 2016. *Nintendo History*. [Online]
Available at: <https://www.nintendo.com/en-gb/Hardware/Nintendo-History/Nintendo-History-625945.html>
[Accessed 11 October 2025].

pyserial, 2022. *pySerial*. [Online]
Available at: <https://github.com/pyserial/pyserial>
[Accessed 9 November 2025].

pyside, 2021. *GitHub - pyside/pyside-setup: Git super project for PySide*. [Online]
Available at: <https://github.com/pyside/pyside-setup>
[Accessed 9 November 2025].

Python, 2020. *socket — Low-level networking interface — Python 3.8.1 documentation*. [Online]
Available at: <https://docs.python.org/3/library/socket.html>
[Accessed 9 November 2025].

Retrocomputing Stack Exchange, 2019. *How does the Gameboy's memory bank switching work?*. [Online]
Available at: <https://retrocomputing.stackexchange.com/questions/11732/how-does-the-gameboys-memory-bank-switching-work>
[Accessed 12 October 2025].

sanni, 2024. *Reading SNES SFC carts..* [Online]
Available at: <https://github.com/sanni/cartreader/wiki/Reading-SNES-SFC-carts>
[Accessed 12 October 2025].

Schneider, J. & Smalley, I., 2024. *What is a microcontroller?*. [Online]
Available at: <https://www.ibm.com/think/topics/microcontroller>
[Accessed 9 October 2025].

SNESdev Wiki, 2022. *Cartridge connector*. [Online]
Available at: https://snes.nesdev.org/wiki/Cartridge_connector
[Accessed 12 October 2025].

SNESdev Wiki, 2025. *SA-1 Pinout*. [Online]
Available at: https://snes.nesdev.org/wiki/SA-1_Pinout
[Accessed 12 October 2025].

SnesLab, 2025. *SA-1*. [Online]
Available at: <https://sneslab.net/wiki/SA-1>
[Accessed 12 October 2025].

Texas Instruments, 2015. *SN74LVC245A Octal Bus Transceiver With 3-State Outputs*. [Online]
Available at:
https://www.ti.com/lit/ds/symlink/sn74lvc245a.pdf?ts=1760160731997&ref_url=https%253A%252F%252Fwww.google.com%252F
[Accessed 11 October 2025].

The Qt Company, n.d. *Qt Features, Framework Essentials, Modules, Tools & Add-Ons*. [Online]
Available at: <https://www.qt.io/product/features>
[Accessed 9 November 2025].

The Qt Company, n.d. *Qt Framework - One framework to rule all!*. [Online]
Available at: <https://www.qt.io/product/framework>
[Accessed 9 November 2025].

TinyGo, 2024. *Microcontrollers*. [Online]
Available at: <https://tinygo.org/docs/reference/microcontrollers>
[Accessed 9 November 2025].

TinyGo, 2024. *TinyGo Home*. [Online]
Available at: <https://tinygo.org>
[Accessed 9 November 2025].

W3Schools, 2011. *C++ Introduction*. [Online]
Available at: https://www.w3schools.com/cpp/cpp_intro.asp
[Accessed 9 November 2025].

W3Schools, n.d. *Introduction to Python*. [Online]
Available at: https://www.w3schools.com/python/python_intro.asp
[Accessed 9 November 2025].

Wikimedia Commons, 2008. *File:Python-logo-notext.svg - Wikimedia Commons*. [Online]
Available at: <https://commons.wikimedia.org/wiki/File:Python-logo-notext.svg>
[Accessed 9 November 2025].

Wikimedia Commons, 2017. *File:ISO C++ Logo.svg - Wikimedia Commons*. [Online]
Available at: https://commons.wikimedia.org/wiki/File:ISO_C%2B%2B_Logo.svg
[Accessed 9 November 2025].

Wikimedia Commons, 2018. *File:MicroPython new logo.jpg - Wikimedia Commons*. [Online]
Available at: https://commons.wikimedia.org/wiki/File:MicroPython_new_logo.jpg
[Accessed 9 November 2025].

Wikimedia Commons, n.d. *Arduino IDE logo*. [Online]

Available at: https://commons.wikimedia.org/wiki/File:Arduino_IDE_logo.svg

[Accessed 9 November 2025].

Yates, J., 2024. *Understanding Buck and Boost Converters and the Capacitors Behind Them*.

[Online]

Available at: <https://blog.knowlescapacitors.com/blog/understanding-buck-and-boost-converters-and-the-capacitors-behind-them>

[Accessed 11 October 2024].

Ziegler, R., 2025. *Gameboy Advance*. [Online]

Available at: <https://reinerziegler.de/mirrors.gg8.se/GBA/gba.htm>

[Accessed 12 October 2025].