Button Remapping for GameCube Games such as Super Smash Bros Melee

Michael Qian, Srikanth Yaganti, Yeda Wu
ECE 445 Proposal - Spring 2020
TA: Evan Widloski

1 Introduction

1.1 Objective

In fighting games, it is usually beneficial to remap certain buttons to perform different actions for ease of doing combos. For example, a player might want to remap the X button on their controller from "jump" to "attack". This is present in the game settings of many popular fighting games except for Super Smash Bros Melee for the Nintendo GameCube.

Our goal is to create an adapter that sits between the GameCube and GameCube controller. The controller will plug into the adapter which plugs into the GameCube. Users will have a phone app where they can choose how to remap their buttons. Users can then load the adapter with multiple button reconfigurations and toggle through these configurations on the adapter. This adapter will then take in the signals of the button presses of the controller and translate them to signal button presses based on the button remapping. This hardware will also allow for button remapping for any other GameCube games.

1.2 Background

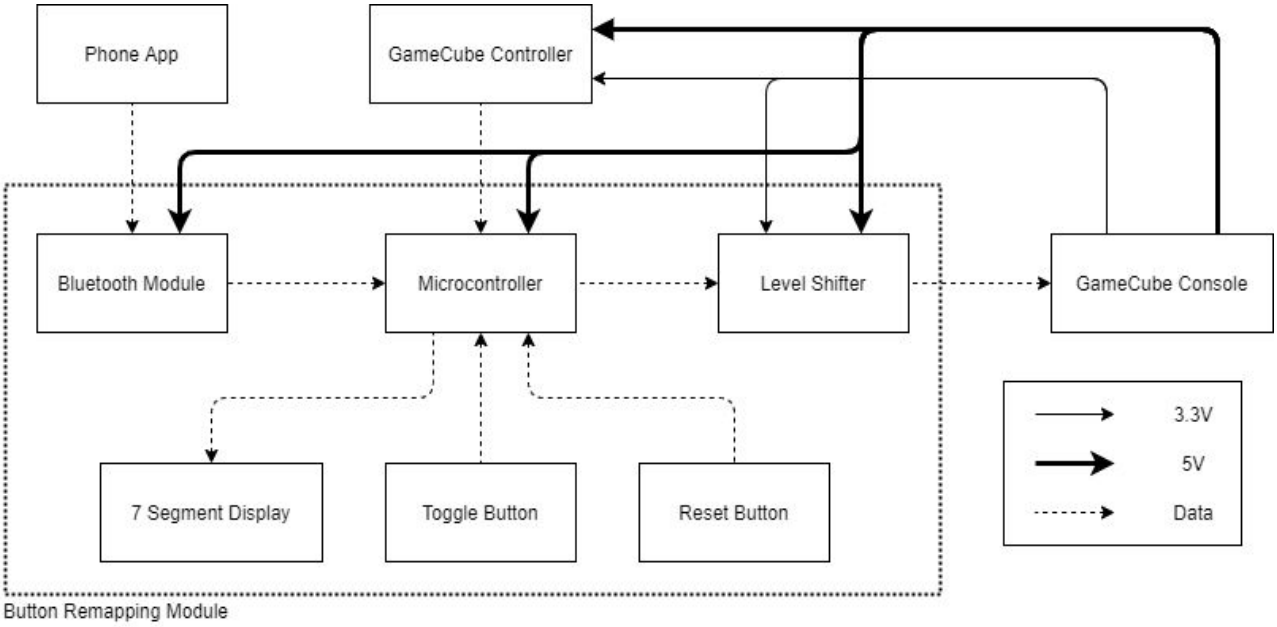
The GameCube came out in 2001[1]. Many third-party controllers came out since then, but only recently have some companies been making controllers which support button remapping. These controllers include the Smash Box[2] and B0XX controllers[3]. Although this is the case, these controllers are not shaped like the GameCube controller. Instead of having analog sticks, they have 4 directional buttons. This causes gameplay to differ from the classic GameCube experience because directional buttons cannot match the precision of analog sticks. Thus, there is still a lack in the market for a hardware component to remap GameCube controller inputs.

1.3 High-level requirements

- Phone app is able to communicate with microcontroller via bluetooth to send the button-remapping schemes
- Microcontroller is able to read the controller inputs
- Microcontroller is able to send the remapped button inputs to the console

2 Design

2.1 Block Diagram



2.2 Functional Overview

2.2.1 Phone

A phone app is required to allow users to easily set button remapping configurations for their controller. This app will connect to the microcontroller via bluetooth.

2.2.2 Button Remapping Module

2.2.2.1 Microcontroller

The microcontroller will convert the input signal from the controller into a remapped output signal that will be driven to the GameCube console. We have determined that the ATmega2560 matches all these criteria and is therefore a good choice for our project.

2.2.2.2 Bluetooth Module

Required for communication with the phone app. It will contain a bluetooth receiver for receiving configurations for button remappings from the phone. We have opted to use the HC-05 Bluetooth module because of its simplicity and compatibility with the ATmega2560.

2.2.2.3 Level Shifter

Since the GameCube console and controller both send data at 3.3V[4] but the ATmega2560 outputs at 5V[5], a level shifter module will be required between the microcontroller and the console to ensure the proper signals are being passed onto the console. We are considering using the TE291 bidirectional logic level shifter.

2.2.2.4 Toggle Button

Since we wish to be able to store and switch between multiple button remapping configurations, a button is needed to cycle through the number of configurations stored within the microcontroller (4 or 8 configurations).

2.2.2.5 7-Segment Display

This will display the current button remapping configuration being implemented by the microcontroller. It is simply a set of LEDs driven by the microcontrollers digital output pins.

2.2.2.6 Reset Button

This button will reset the current configuration and the microcontroller will then simply pass the data from the GameCube controller on to the console without any button remapping.

2.2.3 GameCube Controller

GameCube Controller will communicate with the Microcontroller to send button data. This data will be remapped by the microcontroller and sent to the GameCube console.

2.2.4 GameCube Console:

GameCube Console will receive the proper button mappings from the microcontroller. It will also provide power to all other modules.

2.3 Block-Level Requirement

2.3.1 Phone

A phone app should be able to communicate with the microcontroller via bluetooth. Users should be able to send new button remappings to the microcontroller.

2.3.2 Button Remapping Module

2.3.2.1 Microcontroller

It should be able to send valid data to the GameCube Controller. It will also require enough memory to store the table of button remapped signals. The controller will

need to be configured through Bluetooth and capable of driving the bus to 250kHz to be compatible with the GameCube bus protocol[4].

2.3.2.2 Bluetooth Module

This module must allow the phone app and the microcontroller to communicate with each other. This module must also have a built-in button to perform pairing with the phone.

2.3.2.3 Level Shifter

Level shifter module will be required to convert 5V output from the microcontroller to 3.3V input for GameCube Console. This is crucial to ensure the proper signals are being passed onto the console. It must also be able to drive signals of 250kHz.

2.3.2.4 Toggle Button

Toggle button will be required to smoothly switch and cycle through multiple (4 or 8 configurations) mappings that will be stored on the microcontroller. This button must not cause any mechanical bouncing that would interfere with the signal.

2.3.2.5 7-Segment Display

This display is required to show the current mapping configuration that is being used by the user while using a minimal amount of power.

2.3.2.6 Reset Button

This button is required to reset the configurations to the original GameCube mappings. This button must not cause any mechanical bouncing that would interfere with the signal.

2.3.3 GameCube Controller

The controller will be required to set proper button data to the controller where they will get remapped and sent to the console.

2.3.4 GameCube Console

GameCube Console will be required to properly receive and interpret button data from the microcontroller. It must also be able to provide both 3.3V at 1mA and 5V at 5mA.

2.4 Risk Analysis

There are a few big risks to the success of the project. First is the Bluetooth module being able to properly communicate with the phone app. This risk is two-fold where first, the phone app must be capable of driving Bluetooth communication and then, the hardware Bluetooth module must be able to translate the incoming signal to a format usable by the microcontroller. Second, the communication from the microcontroller to the GameCube console presents a major risk because the console requires a rather specific bus protocol that we must match completely in

order to properly communicate with it. Lastly, the microcontroller we are using is a 100 pin TQFP with a lead pitch of 0.5 mm which will be very difficult to solder and risks producing an electrical short that will prevent the functionality of the project[5].

3 Safety and Ethics

From a hardware perspective, the safety concerns are few but not absent. For example, the GameCube console has a specified signal voltage of 3.3V, whereas the microcontroller and the rest of the modules all require 5V of power. Accidently, miss wiring the power inputs of these two parts could result in the serious damage of both these parts. Furthermore, many of the parts used in this project will be susceptible to electrostatic discharge and thus, precautions must be taken to prevent damaging these parts such as using anti-static gloves and ESD wristbands.

For ethics, we hold responsibility for our project, which is the first rule in the IEEE Code of Ethics[6]. Additionally, our product could introduce a situation within professional gaming if the device is used when non-Nintendo/performance-enhancing devices are not allowed. This would be an unethical use of our device. In addition to this, it may be possible for people to tamper with our microcontroller such that certain button presses can lead to button macros. These two situations violates the IEEE Code of Ethics #9 because, if used in such a way, the trust in and reputation of professional gamers will be harmed[6].

References

[1] “GameCube,” *Wikipedia*, 09-Feb-2020. [Online]. Available: https://en.wikipedia.org/wiki/GameCube. [Accessed: 09-Feb-2020].

[2] “SMASH BOX,” *Hit Box Arcade*. [Online]. Available: https://www.hitboxarcade.com/products/smash-box. [Accessed: 07-Feb-2020].

[3] “B0XX Controller,” *B0XX*. [Online]. Available: https://b0xx.com/. [Accessed: 07-Feb-2020].

[4] “Nintendo Gamecube Controller Protocol,” *Nintendo Gamecube Controller Pinout*. [Online]. Available: http://www.int03.co.uk/crema/hardware/gamecube/gc-control.html. [Accessed: 14-Feb-2020].

[5] “8-bit Atmel Microcontroller with 16/32/64KB In-System Programmable Flash” *ATMEL*. [Online]. Available: https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf. [Accessed: 13-Feb-2020].

[6] “IEEE Code of Ethics,” *IEEE*. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 13-Feb-2020].