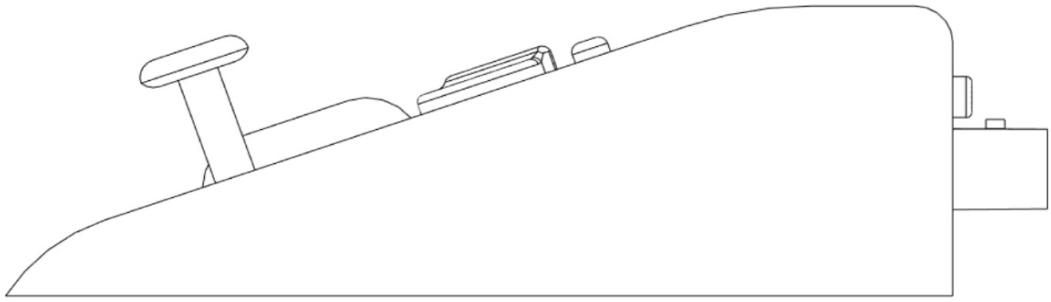
Oscillostation

A retro gaming plug-in console for your oscilloscope

Tanmay Vadhera 2020UEC2553
Mohit Godara 2020UEC2525
Mayank Mathur 2020UEC2501

$17^{th} May, 2022$



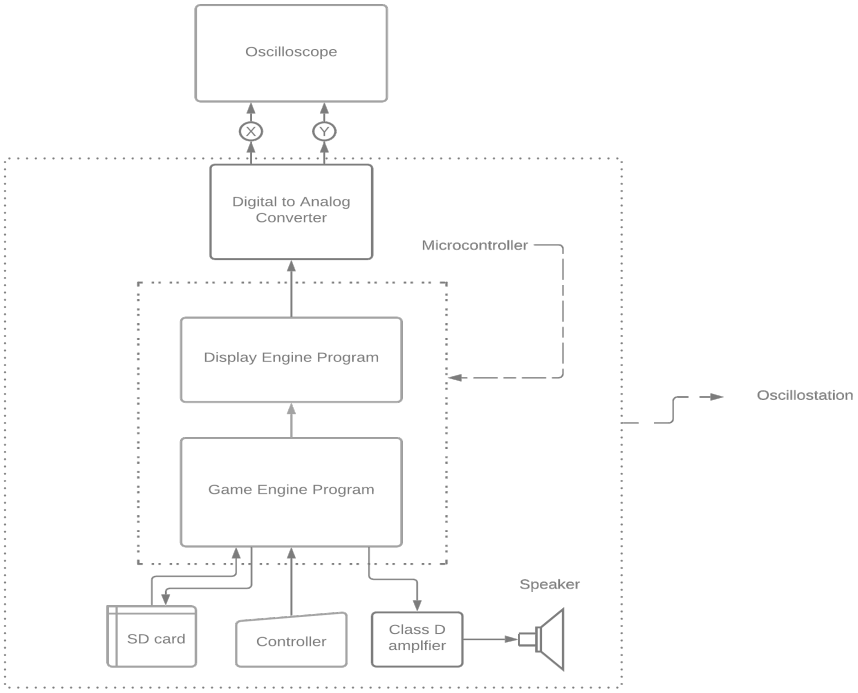
ECECC09 - Electronics Design Workshop
Lab Project
ECE Division
Netaji Subhas University of Technology
Sector-3, Dwarka, New Delhi - 110078

1 Synopsis

For our project,we aim to implement a 2D shooter game on an oscilloscope on its X-Y/versus mode using a STM32 blue-pill, MCP4922 12 bit dual channel ADC, and a few other components. The game is a rendition of the classic space shooter, Space Invaders https://en.m.wikipedia.org/wiki/Space_Invaders .

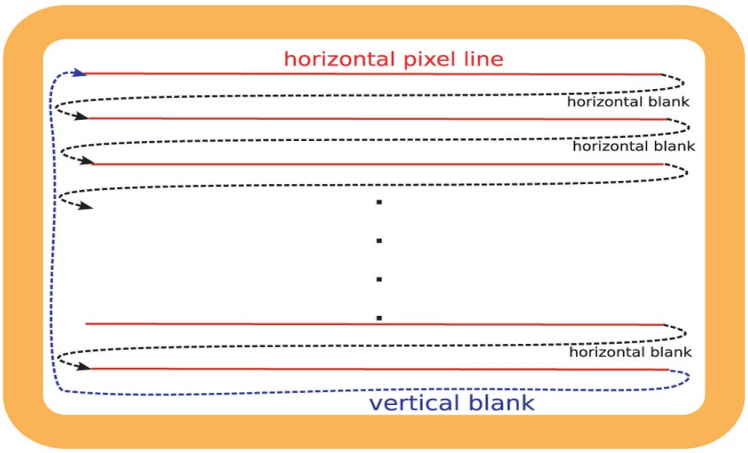
2 Project Overview

The project is divided into two modules: the drawing engine, which consists of the DAC, the MCU code to control it, and the game engine, which consists of a selection of buttons and potentiometers to capture the user's movement to enable them to play the game, view high score statistics, and scroll through and toggle various options.



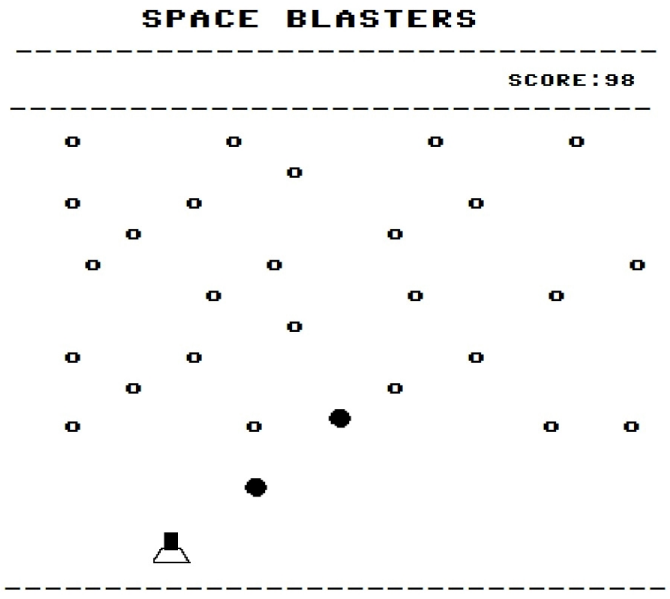
2.1 Drawing engine

The drawing engine relies on the MCU to instruct the DAC using SPI to vary the voltages of its two channels,which correspond to the values of the X and Y coordinates in the oscilloscope's versus mode to plot out the points specified by the game module. Due to a large unspecified number of game elements, a random scanning technique cannot be used, therefore, each frame is rendered using the horizontal raster scanning technique, in which each frame is broken up into pixel wide wide rows spanning the entire horizontal dimension of the frame.



2.2 Game engine

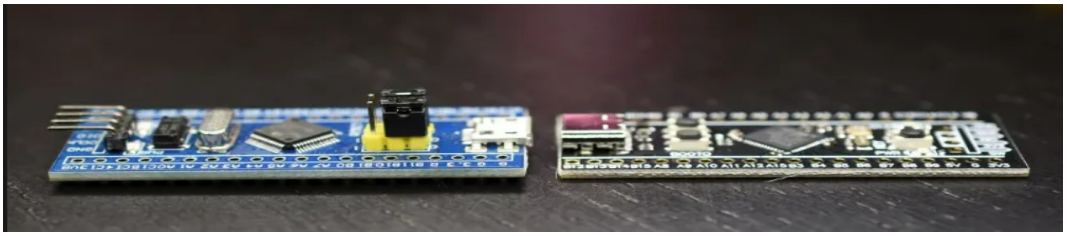
The game engine is also based in the MCU and is responsible for reading and processing the controller inputs. Based on the user's inputs and game logic, the positions of the various game elements such as the turret, moving targets, fired projectiles, score display etc. are updated in each frame. It is also responsible for updating the score statistics in the SD card and allowing the user to view their previous high scores. The game engine will also consist of an 8 bit sound synthesizer to generate various tones to give the user an acoustic feedback while playing the game and scrolling through menus. To accomplish this, the MCU will produce a PWM signal with a duty cycle proportional to the signal of the desired tone. The PWM signal will be fed to a partial class D amplifier consisting of a push-pull power amplifier stage and a filtering stage to filter out the higher harmonics. The output from the filtering stage will be used to drive a 0.5W, 8 Ω speaker.



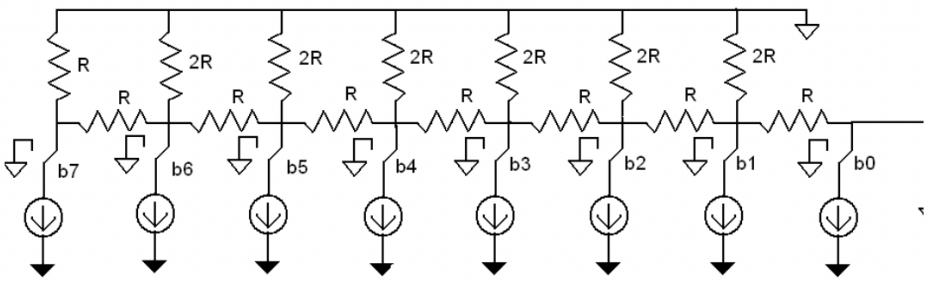
3 Building Blocks

3.1 MCU: STM32 Blue-Pill/Black-Pill

STM32 Blue Pill is a high-performance, breadboard friendly development board with loads of features in a small form factor. It features a 32-bit ARM Cortex M3 processor running at 72MHz frequency with 64Kbytes of flash memory and 20 Kbytes of SRAM. It has an extensive range of I/O and peripherals, including multi-channel Direct Memory Access, several high resolution ADC and three general purpose 16-bit timers plus one PWM timer. It also has multiple communication interfaces including two I2Cs and SPIs, three UARTs, an USB and a CAN. Further, it allows you to add additional flash by soldering and SPI Flash to the board.
The STM32 Blue Pill can be programmed using either the Type-C USB connector (if bootloader is flashed), the ST Link USB dongle, or an external USB to 3.3V TTL adapter.
The Black Pill MCU is pretty much similar to Blue Pill, the only differences are, the USB connector changed from micro USB-B to USB-C, the MCU package is a 48-pin UFQFPN instead of a 48-pin LQFP, we get an extra user button, and the HSE and LSE oscillators are much smaller. The boot mode pins are gone, but we get a boot mode button instead. The black pill MCUs have significantly better specs than the blue pill, with a higher clock speed, more flash storage and more SRAM. A significant change between black pill and blue pill is that the GPIO peripherals were moved off the Advanced Peripheral Bus (APB) onto the AHB.



3.2 8 Bit R2R DAC



The R-2R Ladder is one of the most popular DAC converters because of its simplicity. R-2R Digital-to-Analogue Converter, or DAC, is a data converter that uses two precision resistors to convert a digital binary number into an analog output signal proportional to the value of the digital number. It utilizes only two values of resistors and it can be extended to any number of bits. Furthermore, its output impedance is always equal to R regardless of the number of bits. The number of levels are equal to two power, the number of bits. In this project, 8 bits means there will be 256 levels. We can calculate the maximum voltage output by this equation :

$$Vomax = \frac{Vhigh}{Number\ of\ Levels} * (Number\ of\ Levels - 1)$$

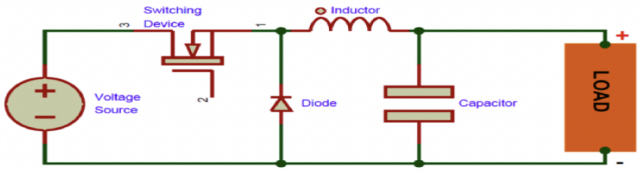
and the voltage output equation for R-2R DAC itself is this :

$$V_{output} = \frac{V_{b0}}{2^n} + \frac{V_{b1}}{2^{n-1}} + \frac{V_{b2}}{2^{n-2}} + \cdots + \frac{V_{bn}}{2^1}$$

In our project here, we made an 8-bit R-2R Ladder DAC, so n equals to 8 and the output voltage can be calculated as :

$$V_{output} = \frac{V_{b0}}{2^8} + \frac{V_{b1}}{2^7} + \frac{V_{b2}}{2^6} + \frac{V_{b3}}{2^5} + \frac{V_{b4}}{2^4} + \frac{V_{b5}}{2^3} + \frac{V_{b6}}{2^2} + \frac{V_{b7}}{2^1}$$

3.3 Buck Converter



The Buck Converter is used in switched-mode power supply (SMPS) circuits where the DC output voltage needs to be lower than the DC input voltage. The DC input can be derived from rectified AC or from any DC supply. It is useful where electrical isolation is not needed between the switching circuit and the output, but where the input is from a rectified AC source, isolation between the AC source and the rectifier could be provided by a mains isolating transformer. The switching transistor between the input and output of the Buck Converter continually

switches on and off at high frequency. To maintain a continuous output, the circuit uses the energy stored in the inductor L, during the on periods of the switching transistor, to continue supplying the load during the off periods. The working of a buck converter can be seen as follows.The switch turns on and lets current flow to the output capacitor, charging it up. Since the voltage across the capacitor cannot rise instantly, and since the inductor limits the charging current, the voltage across the cap during the switching cycle is not the full voltage of the power source. The switch now turns off. Since the current in an inductor cannot change suddenly, the inductor creates a voltage across it. This voltage is allowed to charge the capacitor and power the load through the diode when the switch is turned off, maintaining current output current throughout the switching cycle. This keeps repeating many thousands of times a second, resulting in continuous output.

3.3.1 LM2596 ADJ

The LM2596 series of regulators are monolithic integrated circuits that provide all the active functions for a step-down (buck) switching regulator, capable of driving a 3-A load with excellent line and load regulation. Requiring a minimum number of external components, these regulators are simple to use and include internal frequency compensation, and a fixed-frequency oscillator. The LM2596 series operates at a switching frequency of 150 kHz, thus allowing smaller-sized filter components than what would be required with lower frequency switching regulators.

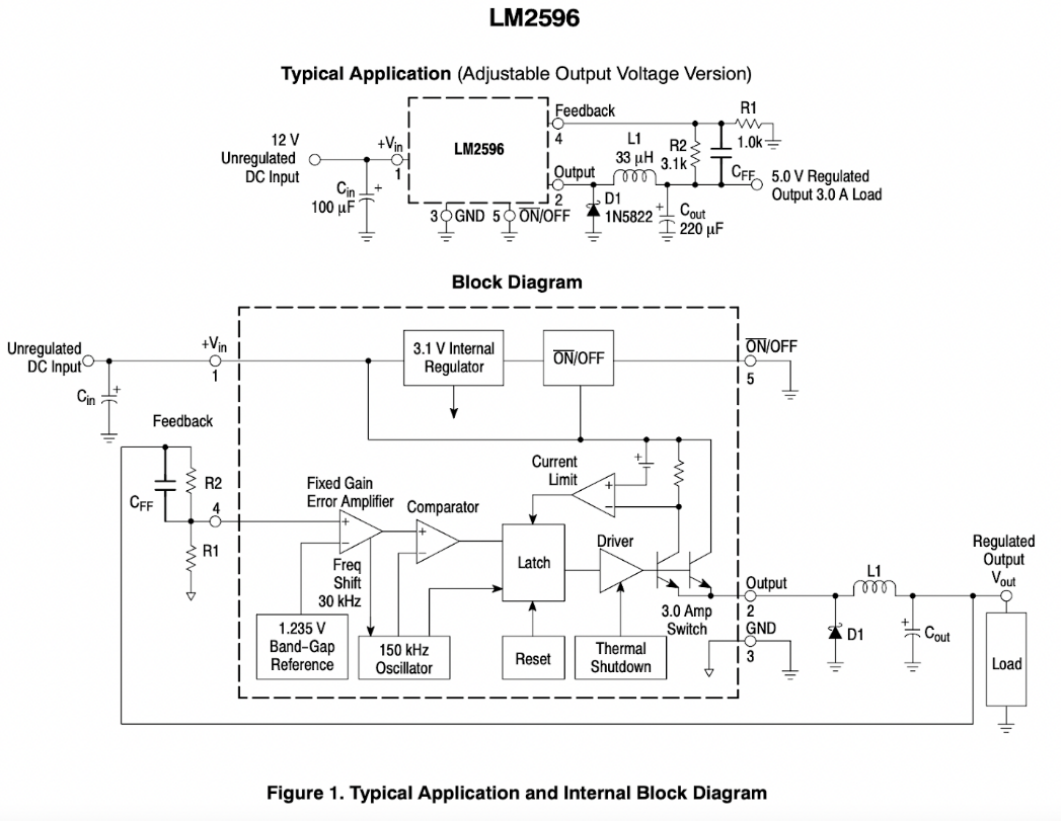
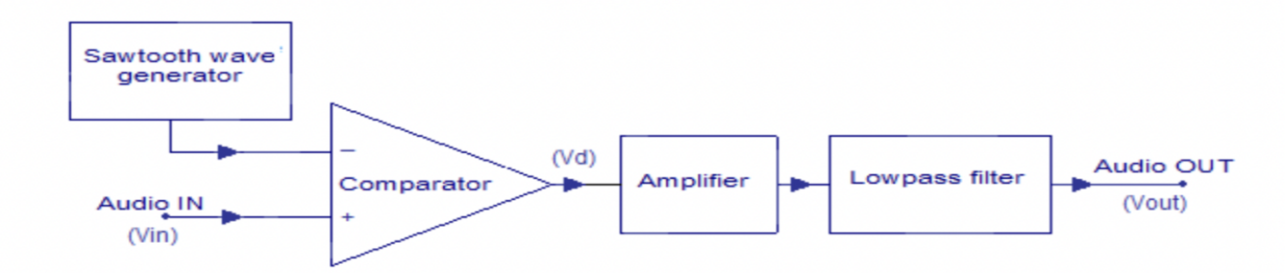
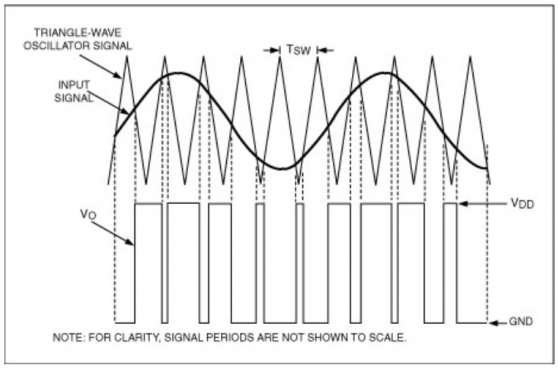


Figure 1. Typical Application and Internal Block Diagram

3.4 Class D Amplifiers

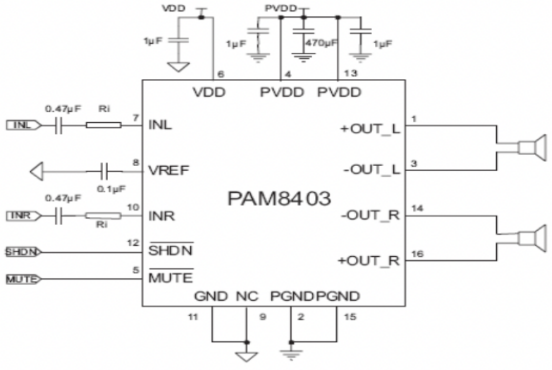
A Class D Amplifier is also known as a 'switching power amplifier'. Class D power amplifier is a type of audio amplifier where the power handling devices are operated as binary switches. Since the power handling devices (MOSFETS) work as perfect binary switches, no time is wasted in between the transition of stages, and no power is wasted in the zero input condition. Class D amplifiers work off pulse width modulation, it converts the input signal into a stream of pulses. Class D Amplifiers are often misinterpreted as digital amplifiers - or at the very least - 'two levels better' than a conventional Class A amplifier. Class D amplifier with real speakers as load will never go below 90% in terms of efficiency. The theoretical efficiency of a Class D amplifier is the ideal 100%.

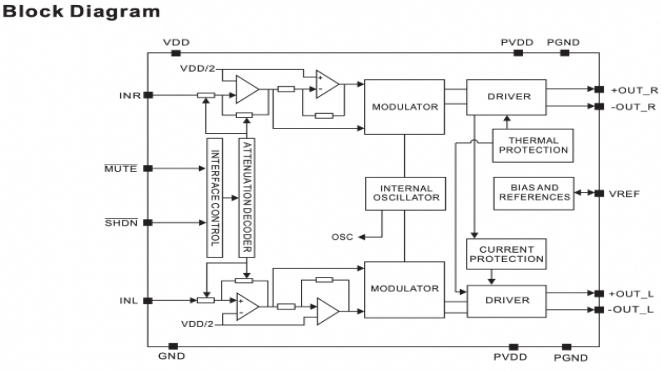




3.4.1 PAM8403

PAM8403 is a 3 Watt, class-D audio amplifier. It offers low THD+N, allowing it to achieve high-quality sound reproduction. The new filterless architecture allows the device to drive the speaker directly, requiring no low-pass output filters, thus saving system cost and PCB area. With the same numbers of external components, the efficiency of the PAM8403 is much better than that of Class-AB cousins. It can extend the battery life, which makes it well-suited for portable applications.
Some of the features of PAM8403 are, 3W Output at 10% THD with a 4Ω Load and 5V Power Supply,Filterless, Low Quiescent Current and Low EMI,Low THD+N,Superior Low Noise, Efficiency up to 90%,Short Circuit Protection,Thermal Shutdown.



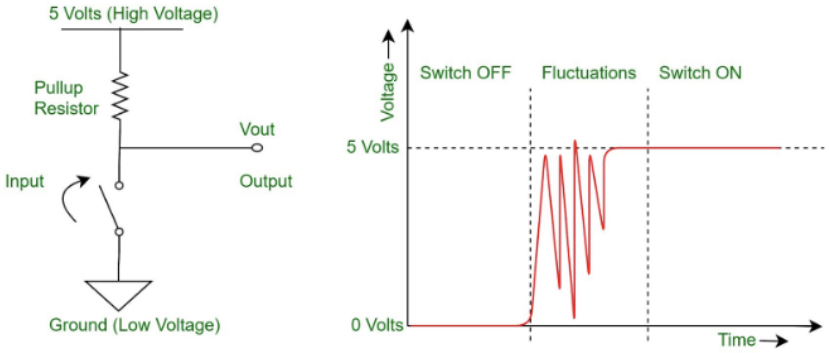


3.5 SPI

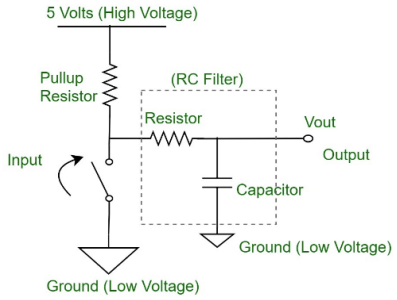
The serial peripheral interface (SPI) allows half/ full-duplex, synchronous, serial communication with external devices. Serial Peripheral Interface (SPI) is an interface bus commonly used to send data between microcontrollers and small peripherals such as shift registers, sensors, and SD cards. It uses separate clock and data lines, along with a select line to choose the device you wish to talk to. The interface is also capable of operating in multimaster configuration. It may be used for a variety of purposes, including simplex synchronous transfers on two lines with a possible bidirectional data line or reliable communication using CRC checking.

3.6 Switch Debouncer

When we press a pushbutton or toggle switch or a micro switch, two metal parts come into contact to short the supply. But they don't connect instantly but the metal parts connect and disconnect several times before the actual stable connection is made. The same thing happens while releasing the button. This results in the false triggering or multiple triggering like the button is pressed multiple times. Simply, we can say that the switch bouncing is the non-ideal behavior of any switch which generates multiple transitions of a single input.
Switch bouncing is a problem, hence, to remove the bouncing from the circuit Switch Debouncing Circuit is used.



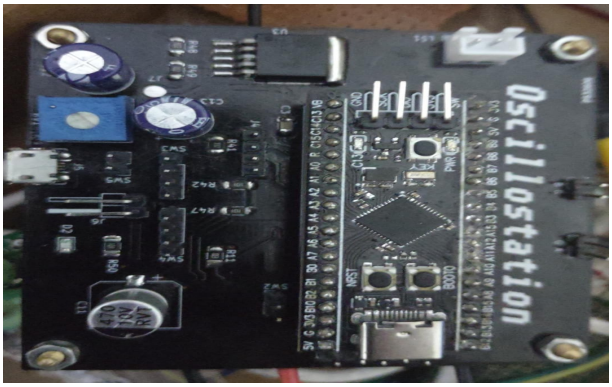
Switch debouncing in an electronic design ensures that the device that is sampling the switch waveform does not misinterpret a single button press as many.
RC FILTER DEBOUNCING
Use of R-C circuit. This circuit involves the combination of a resistor and a capacitor circuit to act as a filter to smooth out the output glitch for the switch. The RC filter slows down the signal transitions generated by the switch so each button press gives one slow edge.

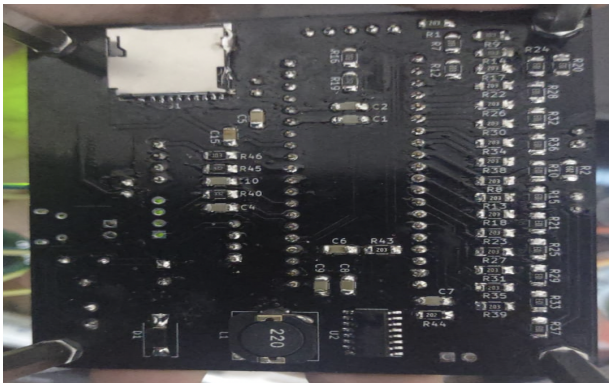


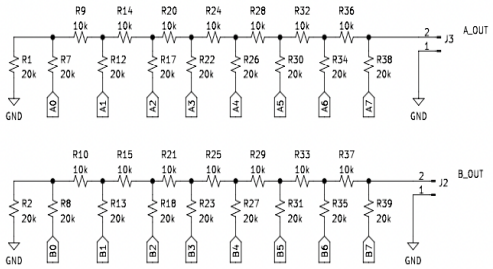
4 Circuit Description

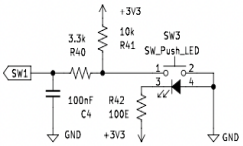
The buck converter section of the circuit handles the conversion of 5V DC input via the microUSB input to 3.3 V DC. This uses a LM2596 ADJ adjustable output buck converter with a maximum load current of 3A.This adjustable version features a drop out voltage of 1.5 V and allows the output voltage to be set via two resistors, which were chosen to be 1.68k (including the rheostat) and 1k for a reference voltage of 1.23V. Other components values were calculated using the LM2596 ADJ design guide present in the datasheet (section 9.2.2,shorturl.at/joxAW). At the centre of the board sits the STM32 blue/black pill, whose pins PB9 to PB15 and PA8 to PA15 are used to drive the 8x2 lines of the two 8 bit R2R DACs. The board also features 5 RC debouncing circuits with a debouncing time of 2ms for the pushbuttons and rotary encoder output and series resistors to drive the built in leds of the pushbuttons. The joystick's analog output gets directly connected to ADC channel 1 and 4 located at pins PA1 and PA4 respectively. A PAM8403 class D amplifier is used to drive a 0.5 W 8 ohm speaker (mono), which gets its

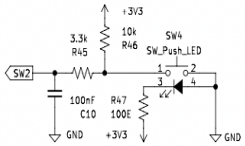
input from pin PB9. Earlier, a RC bandpass filter was used to get a sine wave output, which ,however, was changed in favour of a RC low pass filter (for DC blocking) as the square wave signal had a much crisper sound. The board also features a uSD card slot, which can be used to directly read/write a uSD card using the SPI protocol.

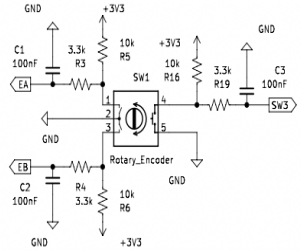


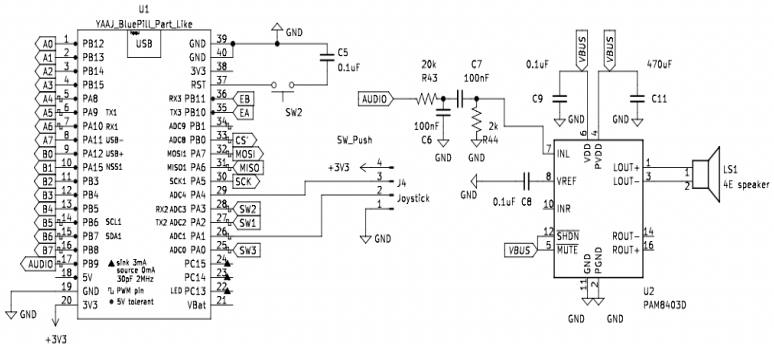


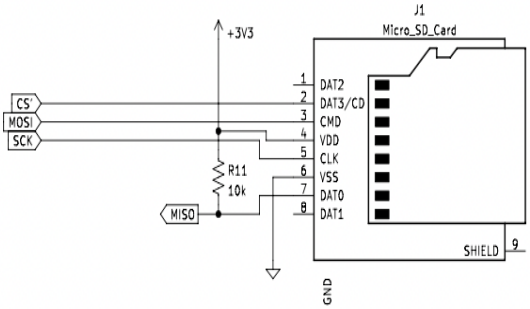


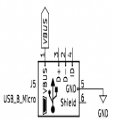


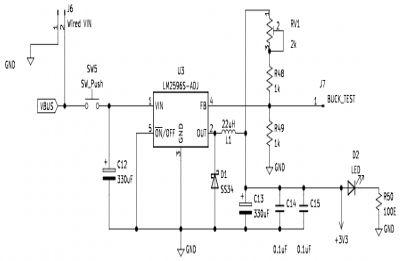












5 Code description

The program starts off by first reading the voltage at PA0 (ADC channel 0). The default 10 bit ADC resolution of analogRead() is used as the onboard 12 bit ADC's full precision is not required. The centre of the turret is

generated by mapping the raw 10 bit ADC values between 20 and 235, using which the outline of the turret is generated (using 3 graphical lines) and loaded into frame_objects matrix (size: n x 5). Frame_objects matrix is used to store the attributes of the objects to be displayed in the general format {object_type(0 for circle, 2 for point to point line),centre_x/xstart,centre_y/ystart,radius/xend,yend}. If the time elapsed since the last target spawn is greater than 15secs, the target generation block is used. First, the radius of the target, which are all circles, is randomly generated between 15 and 25. Then the centre's x coordinate is randomly generated between 30 and 180, this is done so that the number of targets is also randomized, as targets are continuously generated as long as the right most x coordinate of the target is less than 255 (max raw input for 8 bit DAC). Now, the program enters the clean up routine, which deletes targets whose top y coordinate is below the bottom of the screen (0 raw value for DAC) and bullets that have gone above the top of the screen i.e. bottom y coordinate ¿ 255. The routine also checks for any intersections between a target and a bullet and if any is found, a 70 HZ square wave is generated at the speaker to alert the player of a hit and both the objects are deleted. A similar scenario occurs when a collision occurs between the turret and a target, where instead of a tone a melody is played to signify that the game has ended and the program enters an infinite loop. After the cleanup, bullets are generated based on whether the user pushed a button, which is checked using a falling edge hardware interrupt attached to PA2. The bullet generation routine also checks if a cool-off/reloading time of 2.5 secs has passed before entertaining the user's request to fire the turret. The bullets and targets are not directly loaded into the frame objects array as the program moves their y coordinates in different directions and at different speeds across the screen. Once the y coordinates have been updated the mobjs matrix(for targets) and bullets matrix(for bullets) are loaded into the frame_objects matrix using load(). Now the program enters the rendering phase, in which the frame_objects matrix is sorted using the bottom y coordinates of the objects (scanning starts from the bottom) in ascending order using selection sort. After frame_objects is sorted, the program enters a loop in which the scanning takes place starting from y = 0 to y = 255. In each iteration, shapes from frame_objects are checked for intersections with the horizontal line located at the then current y coordinate, this is done by checking whether the bottom y coordinate of the shape is below the current y and the top y coordinate is above the current y coordinate. If the first statement is not satisfied, the scanning of the shapes stops, as the shapes are already been arranged based on their bottom y coordinate and all subsequent shapes will have their y coordinates above the current y. For the second statement, the loop simply skips to the next shape. Further down the loop, the intersection points for the shapes are calculated by calling read_primitives(), which processes the attribute array for the shape and loads the intersection points into the point array. After all shapes are finished scanning, the program heads to the DAC interface section. First, the current y coordinate is used to set the voltage on channel 1 using port A and B's output data registers, which are assigned values based on the pin mappings of the individual bit pins of the 8 bit DAC to the GPIO pins of the STM32 blue/blackpill. Once the y coordinate is set, the x values from the point array are rapidly displayed on channel 2 to generate a single horizontal slice of the frame. The loop continues until the entire frame is horizontally scanned from bottom to top. For the next frame, the positions of the objects are again updated and the cycle repeats.

5.1 Code Reference

5.1.1 GPIO

Stands for "General Purpose Input/Output." GPIO is a type of pin found on an integrated circuit that does not have a specific function. While most pins have a dedicated purpose, such as sending a signal to a certain component, the function of a GPIO pin is customizable and can be controlled by software. Not all chips have GPIO pins, but they are commonly found on multi function chips, such as those used in power managers and audio/video cards. They are also used by system-on-chip (SOC) circuits, which include a processor, memory, and external interfaces all on a single chip. GPIO pins allow these chips to be configured for different purposes and work with several types of components.

5.1.2 ODR

ODR is one of the GPIO registers which hold all output states of pins of a GPIO port. We can alter with 16bit binary value and we specify the pin values with this register. The output data register has 16 bits; the high 16 bits are reserved, and should be preserved as zero (the reset value). If the corresponding pin is configured as an output, the value of each of the low 16 bits is driven onto the pin.

5.1.3 IDR

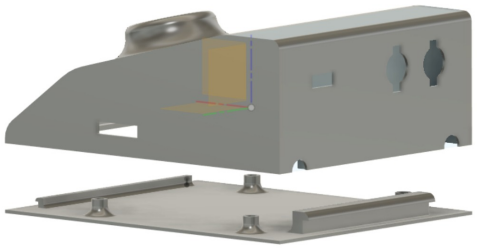
IDR holds all input states of pins of a GPIO port.The input data register has 16 bits; the high 16 bits are reserved, and read as zero. When read, the register returns the logic level present on the pin (after synchronization with the bus clock).

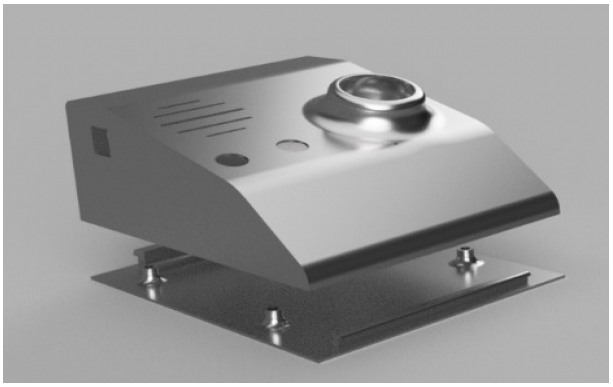
6 Bill Of Material(BOM)

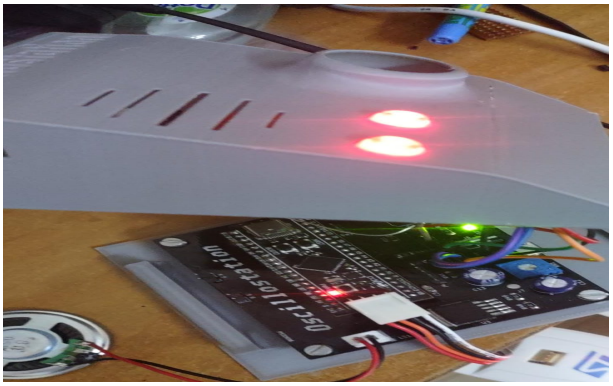
Item	Qty	Reference(s)	Value
1	6	C1, C2, C3, C4, C7, C10	100nF
2	5	C5, C8, C9, C14, C15	0.1uF
3	1	C11	470uF
4	2	C12, C13	330uF
5	1	D1	SS34
6	1	D2	LED
7	1	J1	Micro_SD_Card
8	1	J2	B_OUT
9	1	J3	A_OUT
10	1	J4	Joystick
11	1	J5	USB_B_Micro
12	1	J6	Wired VIN
13	1	J7	BUCK_TEST
14	1	L1	22uH
15	1	LS1	4E speaker
16	18	R1, R2, R7, R8, R12, R13, R17, R18, R22, R23, R26, R27, R30, R31, R34, R35, R38, R39	20k
17	5	R3, R4, R19, R40, R45	3.3k
18	6	R5, R6, R11, R16, R41, R46	10k
19	14	R9, R10, R14, R15, R20, R21, R24, R25, R28, R29, R32, R33, R36, R37	10k
20	2	R42, R47	100E
21	1	R43	20k
22	2	R48, R49	1k
23	1	R50	100E
24	1	RV1	2k
25	1	SW1	Rotary_Encoder
26	2	SW2, SW5	SW_Push
27	2	SW3, SW4	SW_Push_LED
28	1	U1	YAAJ_BluePill_Part_Like
29	1	U2	PAM8403D
30	1	U3	LM2596S-ADJ

7 Enclosure Design

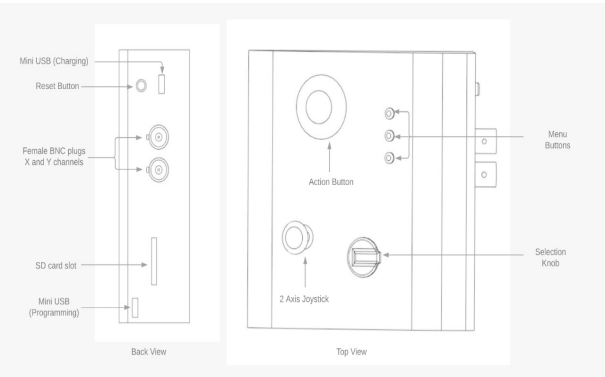
The enclosure is a 3D print in PLA and is designed to mimic the contours of the hand to give the player the best ergonomics during long gaming sessions. The enclosure is based on a clam-shell design and the two halves will be secured using 4 M3 screws. The enclosure also features mounting holes for the various controls and slots for the BNC connectors, SD card ,and the micro USB ports for programming and charging the device.








Oscilloscation


Mini USB (Charging)
Reset Button
Female BNC plugs
X and Y channels
SD card slot
Mini USB
(Programming)
Back View
Action Button
Menu
Buttons
Selection
Knob
2 Axis Joystick
Top View