ELECTRICAL & ELECTRONIC ENGINEERING
STELLENBOSCH UNIVERSITY

DESIGN (E) 314
TECHNICAL REPORT

Audio Recorder and Playback

Author:
Thandi COLLEY

Student Number:
21564027

June 7, 2020

Plagiaatverklaring / Plagiarism Declaration

1. Plagiaat is die oorneem en gebruik van die idees, materiaal en ander intellektuele eiendom van ander persone asof dit jou eie werk is.
   Plagiarism is the use of ideas, material and other intellectual property of another's work and to present is as my own.
2. Ek erken dat die pleeg van plagiaat 'n strafbare oortreding is aangesien dit 'n vorm van diefstal is.
   I agree that plagiarism is a punishable offence because it constitutes theft.
3. Ek verstaan ook dat direkte vertalings plagiaat is.
   I also understand that direct translations are plagiarism.
4. Dienooreenkomstig is alle aanhalings en bydraes vanuit enige bron (ingesluit die internet) volledig verwys (erken). Ek erken dat die woordelikse aanhaal van teks sonder aanhalingstekens (selfs al word die bron volledig erken) plagiaat is.
   Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism.
5. Ek verklaar dat die werk in hierdie skryfstuk vervat, behalwe waar anders aangedui, my eie oorspronklike werk is en dat ek dit nie vantevore in die geheel of gedeeltelik ingehandig het vir bepunting in hierdie module/werkstuk of 'n ander module/werkstuk nie.
   I declare that the work contained in this assignment, except where otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

[illegible]	21564027
Handtekening / *Signature*	Studentenommer / *Student number*
TM Colley	7 June 2020
Voorletters en van / *Initials and surname*	Datum / *Date*

Abstract

The simple audio recorder discussed in this report is a proposed solution to the user who needs an inexpensive device with which to record and play back their own audio. The device makes use of an STM32 microcontroller and is powered internally by a 9V battery. The user controls the recording, playback and volume of the device by use of a simple user interface. The device is compatible with any microSD card and stereo headphones and is compact and lightweight.

Contents

List of Figures

List of Tables

List of Abbreviations

ADC	analog-to-digital converter
CLK	clock
CS	chip select
DAC	digital-to-analog converter
FATfs	file allocation table file system
FSM	finite state machine
GND	ground
GPIO	general-purpose input/output
IC	integrated circuit
IDE	integrated development environment
LED	light-emitting diode
MCU	microcontroller unit
MISO	master-in, slave-out
MOSI	master-out, slave-in
PCB	printed circuit board

PDD	project definition document
SD	secure digital
SPI	serial peripheral interface
TIC	test interface connector
TX	transmission
UART	universal asynchronous reciever/transmitter
UI	user interface

List of Symbols

I_{cap}	capacitive current
n	chosen recording slot (1, 2 or 3)
T	counter period, auto reload value
Vpk	peak voltage value of a sinusoidal signal

1 Introduction

This report examines the design process, fabrication and testing process for a basic audio recorder. There are various requirements that need to be met, in accordance with the Project Definition Document (PDD) [1]. The recorder must be able to record three separate audio files, with a maximum length 20 seconds each to an SD card, and play them back upon request. The recording, playback, and ability to stop either of these actions is made possible by a user interface configuration of pushbuttons. A series of LEDs indicate the state of the recorder and which slot is being recorded to or played back from.

The STM32 microcontroller unit is the main interfacing channel of the recorder. It connects various peripherals, including a microphone module, SD card module the user interface, and the audio output channel. The system is externally powered by an unregulated 9V supply, which is regulated down to 5V and 3.3V respectively to power the microcontroller and peripherals.

A Test Interface Connector (TIC) is used to demonstrate and assess the functionality of the recorder. A direct connection is made between the microcontroller (and various peripherals) and the TIC, which simulates user input and tests the results.

The design process for the hardware and software functionality of the recorder is discussed in detail in this report. Various tests were conducted to ensure that the project requirements are met, and the details and results of these tests are covered in the report. Not all tests were able to be formally conducted equipment access was limited after the national Covid-19 lockdown. However, the test methods and predicted results will be reported on. Finally, a short discussion on the design shortcomings, non-compliances, and any recommendations is included.

2 System description

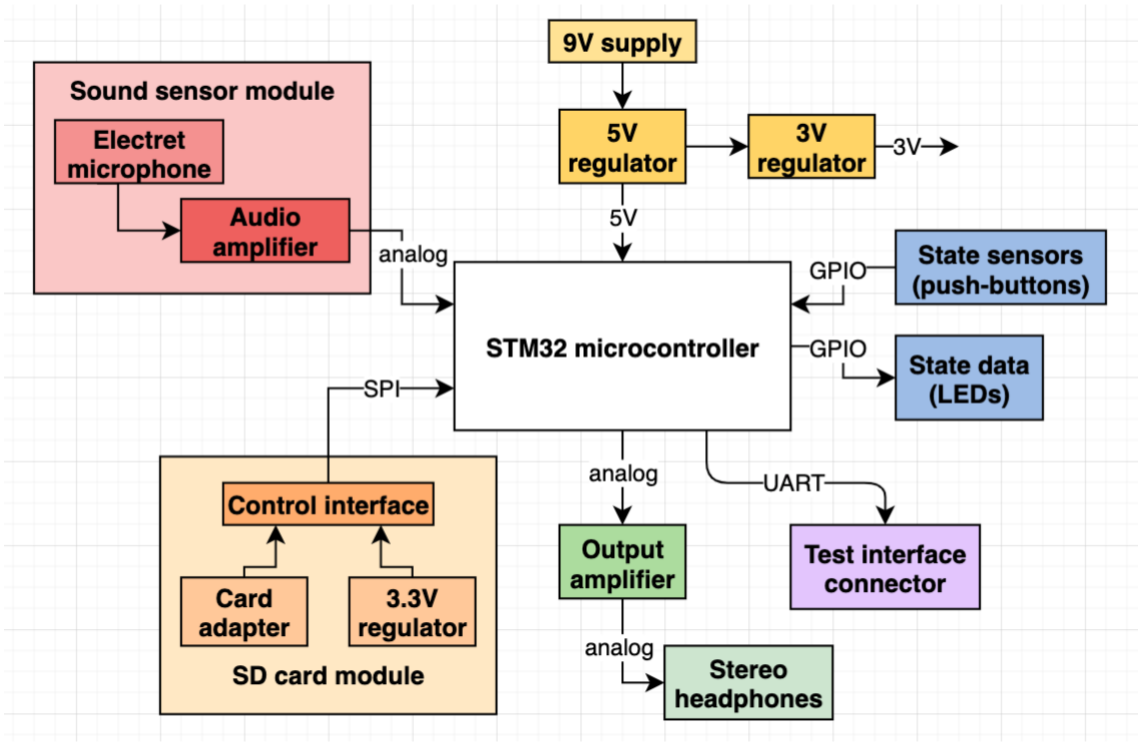

Figure 1: System block diagram

2.1 Microcontroller

The STM32 Nucleo-F446 development board was selected for this system. The STM32 Cube IDE is used to manage the peripheral interfacing and program code for the system. Figure 1 shows how the different components interact with the board and through which interface connections. Table 3 in Appendix F details the pin configuration of the MCU.

2.2 Power supply

The system is designed to receive 9V external power, which is regulated down to 5V and then to 3.3V in series. Table 1 shows the voltages supplied to each component.

Table 1: Component voltage supply

Component	Operating Voltage
STM32 microcontroller	5.0 V
Pushbuttons	3.3 V
Output amplifier	3.3 V
Sound sensor module	3.3 V
SD card module	5.0 V

2.3 UART communications

UART communication is used to transfer debug information from the MCU to the TIC. It does not add any functionality to the system, but allows for confirmation of system startup, an indication of system state changes, and can analyse the audio data points if necessary.

2.4 User interface

The user can interact with the system using a combination of pushbuttons switches and LEDs on the system baseboard. The pushbuttons are used to specify the behaviour of the system, and are able to differentiate "record", "playback" and "stop" functionality. The LEDs indicate the active state of the system. Each system state is described in Table 2 below. These user interface components are connected to the microcontroller via the GPIO interfacing pins on the development board, as detailed in Table 3 from Appendix F. Power is supplied to each pushbutton from the 3.3V regulator.

Table 2: User interface

State	Button combination	LED feedback
Record to slot *n*	Push and hold "rec" button; push button *n* and release to start recording	LED "rec" on; LED *n* flashes for 250ms on and 250ms off, while recording takes place
Play back recording from slot *n*	Push button *n* and release to start play-back	LED *n* flashes for 250ms on and 250ms off, while playback takes place
Stop recording/playback	Push button "stop" and release to stop recording or playback	All LEDs off

2.5 Microphone input

The Waveshare Sound Sensor module was used as an audio input to the system. The module features an in integrated electret microphone and pre-amplifier and is directly connected to the ADC input pin of the development board. Power is supplied to the module from the 3.3V regulator. When the system is in its "record" state, the analog data from the sound sensor will be sampled at a rate of 44.1kHz and processed for storage and playback.

2.6 Audio output

The sampled audio is translated back to an analog signal by the DAC of the microcontroller. This analog signal from the DAC is amplified by op-amp circuitry on the system baseboard, and then sent to a stereo socket, which standard headphones can connect to in order for the audio to be listened to. This headphone load is modeled as a 16Ω resistor. There is a direct connection between the DAC interface of the microcontroller and the op-amp. The op-amp circuit is powered by the 3.3V regulator.

2.7 SD card

A Catalex MicroSD Card Adapter module is connected directly to the SPI interface of the microcontroller. The module features a card socket, into which a SanDisk 16GB SD card (or any microSD card) can be inserted. The power supplied to the module is 5V, which is regulated down to 3.3V by the integrated power regulator on the adapter. Three separate audio files, with a maximum duration of 20 seconds each, can be stored on the SD card at any given time. The "record" functionality of the system writes audio files to the card, and the "playback" functionality reads selected audio files from the card.

3 Hardware design and implementation

3.1 Power supply

Figures 5 and 6 in Appendix A depict the schematic and circuit diagram for the power supply. An L7805CV voltage regulator converts the unregulated DC input voltage, more or less 9V, to 5V. That 5V is connected via the base board PCB and separate wires to power the Nucleo-board, and the SD card module. The 5V is also further regulated down to 3.3V by the MCP1700 voltage regulator. This is the operating voltage for most peripherals used, as stipulated in section 2.2.

Both of these voltage regulators were manually soldered onto the baseboard and use separate wires to connect to the peripherals. Ceramic decoupling capacitors of 0.1μF and 1μF are used to increase the stability of the power supply, as well as a 1N4007 1A rectifier diode at the input. *LED0* is used as an indication that power is on. The power supply circuit followed the typical application circuit as depicted in the MCP Low Quiescent Current LDO datasheet [3] and the LM78XX / LM78XXA 3-Terminal 1 A Positive Voltage Regulator datasheet [2], as well as what is recommended in the Power Supply Connections circuit diagram in the Product Definition Document (PDD) [1]. These recommendations include the values for C1, C2, C3, Cin, Cout and R12, as depicted in Figures 5 and 6 in Appendix A. These values were appropriate in range for this project, and so no further design was needed.

3.2 UART communications

UART data transmission is sent straight from the microcontroller to the Test Interface Connector (TIC). A wire connection is made from the CN3 transmission (TX) pin of the microcontroller and soldered to pin 8 of the TIC, as can be seen in Tables 3 and 4 in the appendix.

3.3 Buttons

Four push-button switches are used for the user interface as stipulated in section 2.3. They are soldered onto the baseboard onto designated positions. Each switch is directly connected to the 3.3V power supply, as well as a GPIO connection to the microcontroller. We enable a pull-down resistor via the STM IDE in order to pull a button which is not pressed to ground, so that a false high state is not read from it. Figure 2 depicts the physical layout of the buttons, as well as the PCB headers, GPIO pins and TIC pins associated with each button.

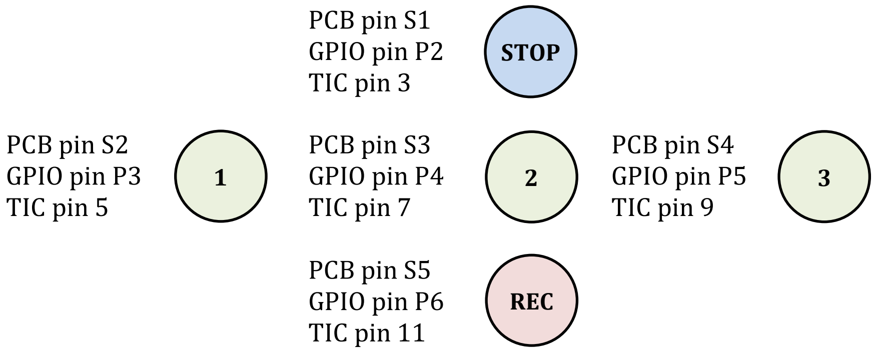

Figure 2: Push-button layout and connection description

3.4 LEDs

The LEDs are soldered onto the project baseboard at PCB positions D2 – D5, depicted in Figure 7 in Appendix B. Each LED indicates a certain state of the recorder, as stipulated in section 2.3. The LEDs are connected directly to the GPIO pins of the microcontroller, which will send an output signal of 3.3V to the LED in order to turn it on. Each LED also has a direct connection to ground, and the TIC. A 1kΩ resistor is soldered in series with each LED, since the LEDs would break if the entire 3.3V GPIO output signal was connected across it. The resistor lowers the effective voltage across the LED to a more reasonable working level.

3.5 Microphone

The Waveshare Sound Sensor module is used for the project. This module can be directly soldered onto the project baseboard. The audio output is sent to be processed by the microcontroller ADC, but the output signal of the microphone cannot directly connect to the ADC of the MCU, since its amplitude is far too small and has negative values. The integrated pre-amplifier on the module board fixes this amplitude and biases the signal around half the supply voltage, as noted by the amplifier datasheet [4]. For a supply of 3.3V, this will make the zero-point around 1.65V, which is in the optimal range for the ADC. Aside from the 3.3V power supply, a direct ground connection and a direct line to the microcontroller ADC, no additional hardware design is needed. It is possible to adjust the signal gain by using the variable resistor on the module, but this did not prove necessary for the project.

3.6 Audio output

The sinusoidal DAC output from the MCU needs to be processed before it is ready to be sent through the stereo socket. The configuration in Figure 10 in Appendix E is soldered onto a prototyping area of the project baseboard. The 1kΩ resistor, in combination with a variable resistor, lowers the amplitude of the signal – this can be seen as a volume dial. Thereafter, the signal is sent through the MCP603 op-amp, which acts as a voltage follower and buffer, drawing current from the 3.3V power source instead of the MCU. The 10μF capacitor removes the DC offset which is a result of the fact that the DAC cannot provide a negative voltage.

The generated audio signals will ultimately have a peak voltage of 200mV and be centered around 0V, as required by the PDD [3]. There is also a direct connection from the audio output signal to the TIC.

3.7 SD card

The Catalex MicroSD Card Adapter is the interface of choice for communication between the MCU and our chosen microSD card (SanDisk 16 GB). MicroSD was chosen since it is physically smaller and therefore easier to implement than standard SD. Since SPI mode is being used as opposed to SD mode, interfacing is much easier, as the MCU already has an SPI peripheral built in.

The card adapter module features a card socket, a voltage regulator circuit, a logic buffer and a control interface. The voltage regulator is necessary because even though the module uses 3.3V logic levels, it assumes a 5V input, and so although we do have access to a 3.3V power supply, we need to make a connection to the 5V line so that the module functions correctly.

The control interface allows us to connect from the MCU directly to the MOSI, MISO, CLK and CS pins of the MCU. A VCC and ground connection also need to be made.

4 Software design and implementation

4.1 High-level description of program

The MCU has various roles to fulfil on a high level. On startup, all relevant peripherals and variables are initialised. It recieves the user interface pushbutton input and consequently assigns a system state and implements the appropriate behaviour in order to achieve the desired output. The functionality required to record and store audio data using the microphone and SD card modules is included. It also processes the data and transmits it to the audio output. Debug information and audio data is sent to the TIC via UART communication.

4.2 Control logic

The finite state machine (FSM) of the system in Figure 3 was implemented programmatically. At startup, all of the peripheral functions and global variables are initialised. The program then enters an infinite `while()` loop and waits in idle for an interrupt from the UI in the form of a button press. Since there is a specific combination of button presses needed to start a recording, there is an intermediate state, "expect recording", wherein the user could initiate recording to a slot *n* or release the "rec" button to effectively cancel the recording. Both the recording and playback functions are only initialised upon the release of the button *n*, and so the button needs to be configured in falling-edge trigger mode.

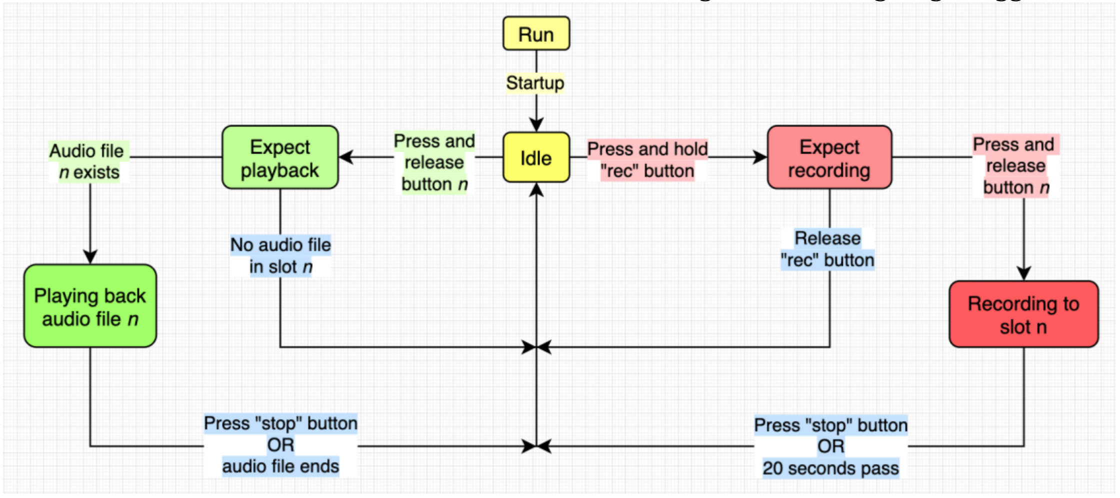
Run
Startup
Idle
Press and release button *n*
Expect playback
Audio file *n* exists
Playing back audio file *n*
No audio file in slot *n*
Press and hold "rec" button
Expect recording
Press and release button *n*
Recording to slot n
Release "rec" button
Press "stop" button OR audio file ends
Press "stop" button OR 20 seconds pass

Figure 3: System finite state machine

4.3 Button bounce handling

It is possible for false input data to be received as a result of non-ideal mechanical behaviour exhibited by the pushbuttons upon pressing. This can be combatted by checking that an interrupt signal coming from a pushbutton press is more than a certain debounce period later than the previous one, before being interpreted as intentional and continuing with the execution of the action. In order to implement this, the time at the start of each button press interrupt is saved into a variable (using the `HAL_GetTick()` function) and compared to the time recorded during the previous interrupt. 10ms has been chosen as the debounce period for this system, and after testing proves to be a suitable amount of time to prevent the effects of button bounce.

4.4 Data flow and processing

Once the sound sensor module has received analog audio data, it is sent to an ADC interface pin on the MCU. On the IDE, the ADC data resolution is set to 8 bits, in order not to generate too much data for the limited SRAM. This means that there will be one byte of data per sample, which also makes processing the audio simpler from a design perspective. A timer is used to trigger the ADC conversion, at a sample rate of 44.1kHz. The ADC process also makes use of DMA, for which the DMA settings need to include the additional ADC at a data width of one byte. For this project, circular mode DMA was selected, so that the conversion automatically restarts after each cycle. A UART channel also needs to be added using the DMA settings.

A low-pass filter is implemented to remove noise from the audio signal, and the DC offset is removed so that the signal centres around zero. After processing, the ADC sample values are stored into a DMA buffer which has 1024 elements. This buffer is used to transmit debug data to the UART, using the `HAL_UART_Transmit_DMA()` function inside of a half- and complete-callback function combination.

4.5 SD card interfacing

The order of communication for the SD card functionality, as simplified in the instructional video on SD Card Low-Level Interfacing [5], is shown in Figure 5.

The system executes the functions used to mount, open, read and write to the SD card from within the main program. Within the FAT file system functions, the disk is initialised using the `disk_initialize`, `disk_read` and `disk_write` functions from the `diskio.c` file, which calls the SD card driver functions. Low-level SPI interaction is implemented by the SD card driver using HAL SPI functions, which interact with the SPI hardware.

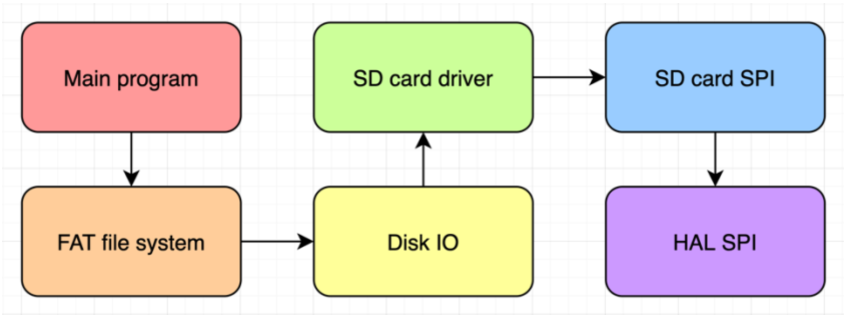

Figure 4: SD interfacing software layers

4.6 Peripheral setup

A description of the pin configuration between the MCU and various peripherals is given in Section 7.2. The peripheral setup done via the STM32 Cube IDE for the ADC has already been examined in Section 4.4.

The pushbuttons are each configured as *GPIO_Input* connections and feature an activated GPIO pull-down resistor as discussed in section 3.3. All the LEDs have *GPIO_Output* connections.

An internal clock source, *TIM2*, is enabled in update-event trigger mode. The counter has a frequency of 84MHz, and since we are aiming for a sample/playback rate of 44.1kHz we choose a prescaler value of 0 and a counter period of 1095.

$$T = \frac{f}{BR} = \frac{84000000}{44100}$$

$$\therefore T = 1905$$

A UART channel, *USART2*, is enabled in asynchronous mode with a word length of 8 bits at a bit rate of 500 000 bits per second. There is a communication stream between the UART data and the DMA, which is configured in the memory-to-peripheral direction. Another memory-to-peripheral DMA stream is used for the DAC, which transmits our analog audio data.

Various separate SPI connections are made for the SD card interface. *SPI2* is used in full-duplex master mode to manage the data transmission of the *SPI2_CLK*, *SPI_MOSI* and *SPI_MISO* signals

5 Measurements and Results

As a result of the nationwide Covid-19 lockdown, many of the measurement and testing techniques described below were not possible for this project, since not all components were available and access to the necessary lab equipment was limited. In an ideal case, the measurement values would have been tabulated so as to directly compare them to the theoretical values, and the error margin would have been indicated. There are various factors such as non-linearities, unpredicted behaviour and more which should have been covered in this section but unfortunately can no longer be discussed. However, the techniques and tests described would sufficiently deliver adequate measurements which should warrant further conclusions, had lab access been possible.

5.1 Power supply

A multimeter in DC-mode can be used to check for the correct values and for continuity at each stage of the power supply, as well as at the supply nodes of each peripheral. To test that the voltage level is correct, connect one pin of the multimeter to ground, and the other at the node in question. The multimeter should display the desired value (5V or 3.3V etc.).

In order to test for continuity, set the multimeter to continuity-mode and connect the pins to either end of the node being tested. The multimeter should beep if there is electrical continuity. This continuity test should be performed for all peripherals discussed below in order to ensure proper connectivity throughout the system.

5.2 UART communications

UART communication can be tested using an oscilliscope probe connected at the *TX* pin of the MCU. The oscilliscope should display the binary data values of the UART output. The binary values can be translated to ASCII values, which should indicate the startup message and any state changes that occur in the system. This information can be used to debug the device.

5.3 Buttons

Button functionality can be tested with an oscilliscope to ensure that the voltage reading at the switch output reads high (3.3V) when the button is pressed and is zero when the button is not pressed.

In order to test whether the GPIO peripheral interfacing was done correctly on the IDE, a simple function can be written which simply turns one the the UI LEDs when the button is pressed and turns the LED off when the button is released. If the LED does not turn on whe the button is pressed, then there might be an interfacing error or a hardware error.

5.4 LEDs

LED functionality is fairly easy to test as it is in essence a visual indicator of its own state. Before the system state machine was written, a simple *GPIO_Out* function was written for each LED to simply turn on at startup, in order to see if the hardware design and peripheral setup were correct.

In the process of designing this project, there was an issue whereby one of the LEDs was not flashing when expected to, even though the others were. The program logic was sound, so a hardware test, using a multimeter for continuity, was conducted. It was discovered that the connection on the PCB between the resistor and LED was broken, and so a hardware hack was done in order to make a new physical connection between the two components. Thereafter, the LED functioned as expected.

5.5 Microphone

An oscilloscope can be used to observe the microphone signal at the *AOUT* pin of the module. When there is no audio input, the measured voltage should be a constant value above zero (as a result of the amplifier IC). There should be irregular sinusoidal-like signals observed by the oscilloscope if sounds like talking or music are captured by the microphone.

5.6 Audio output

The DAC output signal can be measured using an oscilliscope probe. The probe needs to have a x10 attenuation in order to ensure that there is a high enough impedence so that the measurement is accurate. The oscilliscope measurements will vary depending on whether a load (pair of headphones) is connected or not, so be sure to investigate both scenarios. With a 16Ω load, the measured signal at the output of the amplifier should be centred around zero and have a V_{pk} value of 200mV.

The equation below illustrates that a small distortion at the DC blocking capacitor input can have a significant effect on the output. Therefore, it is important to ensure that the amplifiier output does not exhibit clipping when the load is connected. A distorted signal can be adjusted by reducing the amplitude of the DAC signal in the `wave_init()` function program code.

$$I_{cap} = C\frac{dV}{dx}$$

5.7 SD card

Each individual signal from the SD card can be checked for working functionality. For the CS signal, a multimeter can be used to confirm that the signal goes low and then high. An oscilliscope can be used to observe the SPI signals, like the required pulses of the clock signal at the CLK pin, as well as data input and output waves at the $MOSI$ and $MISO$ pins respectively.

In order to test that data has been written to the card, it can be insterted into a computer and the files can be opened in any audio software program, such as Audacity. The recorded files can be inspected and played back this way, to confirm that the SD interfacing is occuring correctly.

The observed results of the stored files will vary in accordance with the processing code as stipulated in Section 4.4. These testing methods will allow us to adjust the filter code, sample rate and more to best suit the project requirements.

6 Conclusions

6.1 Non-compliances

All requirements for the project (in accordance with the PDD [1]) were met upon formal demonstration of the power supply, UART communication, buttons, LEDs and audio output. The only exception was an error in the state machine which did not turn the “record” LED on when it was supposed to. This was due to a timing error which could not be further investigated after lab access was limited.

As discussed in the preamble of Section 5, any further non-compliances which may have occured were not observed since the system was not able to be properly tested. This includes functionality for the microphone and SD card.

6.2 Design shortcomings, recommendations and possible improvements

This recorder is a fairly fragile system as the hardware components and circuitry are uncovered and therefore more likely to be damaged by external disturbances. This can be improved upon by creating a durable cover for the PCB which allows for the necessary peripheral interfacing and UI capability but is able to protect the other components of the system. Such a cover could also improve the ergonomic “feel” of the device, as stipulated in the user requirements of the PDD [1].

Another user requirement for the project is that it is powered by an internal 9V battery, which is not accounted for in the final design. Currently, the power is supplied by an external DC connector. Adjustments could easily be made to the board layout to accommodate an internal battery, and very few connections would need to change. In fact, the project is already compatible with an external 9V bettery connected in a rudimentary fashion using the recommended red and black power cables instead of a laboratory voltage generator.

Functionality could be added to this project to allow more audio files to be recorded and played back. The control logic is certainly scalable, and so as long as the system memory is capable, this should be trivial implement. Similarly, the maximum length of the audio file could be easily be increased.

The volume control functionality was not properly investigated as a user control option. The variable resistor in the audio amplifier does behave as a volume adjuster, but it is a difficult component to physically adjust. In future builds, this could be accommodated to add extra functionality and comply with the user requirement as stipulated by the PDD [1].

7 Appendix

7.1 Schematics

7.1.1 Appendix A: power supply

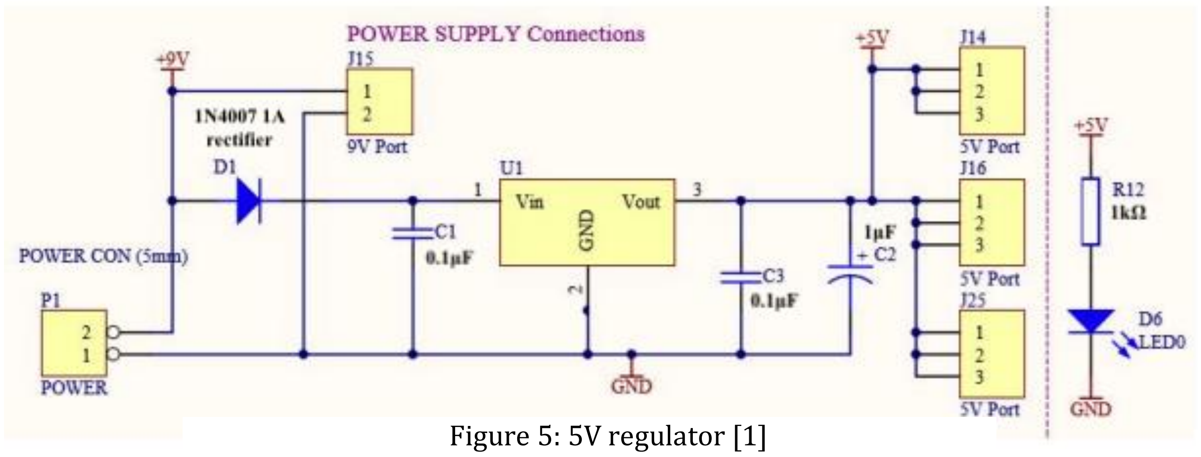

Figure 5: 5V regulator [1]

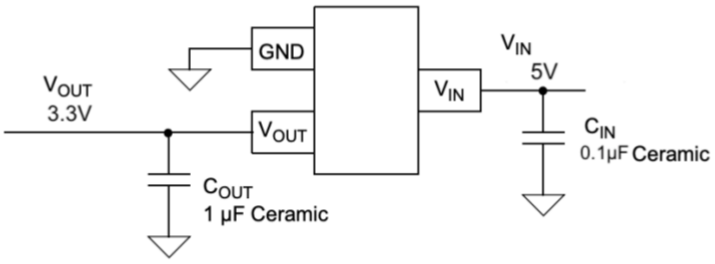

Figure 6: 3V regulator [3]

7.1.2 Appendix B: user interface

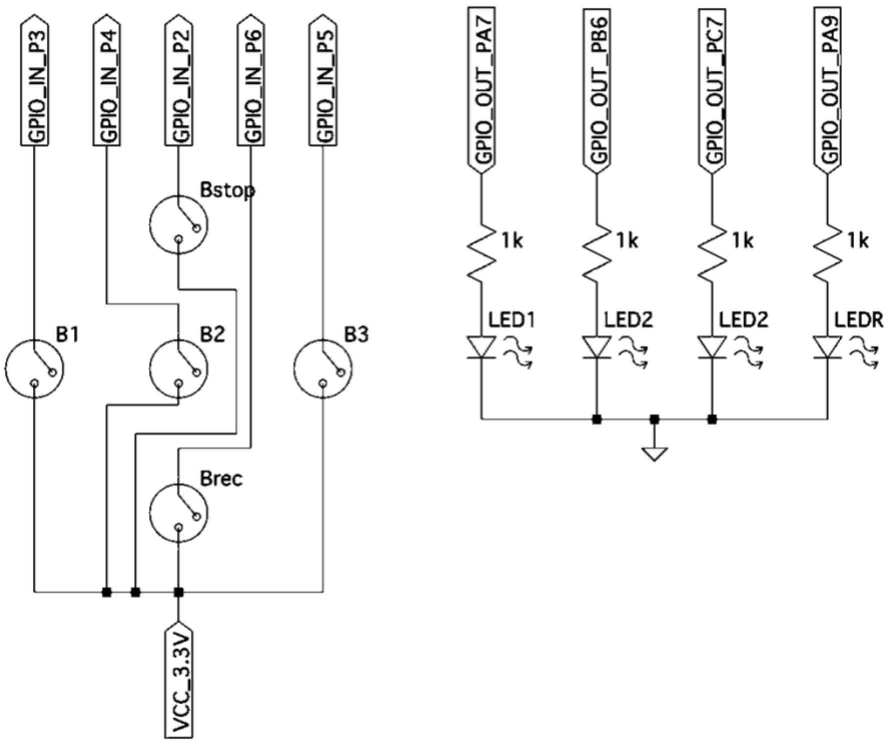

Figure 7: User interface

7.1.3 Appendix C: microphone module

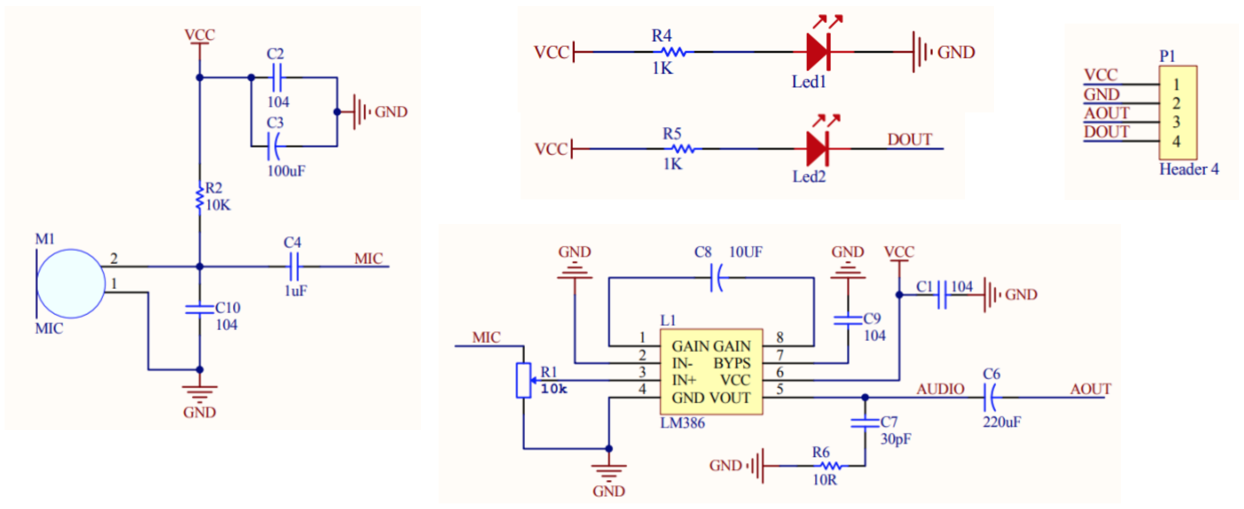

Figure 8: Microphone module [6]

7.1.4 Appendix D: SD card module

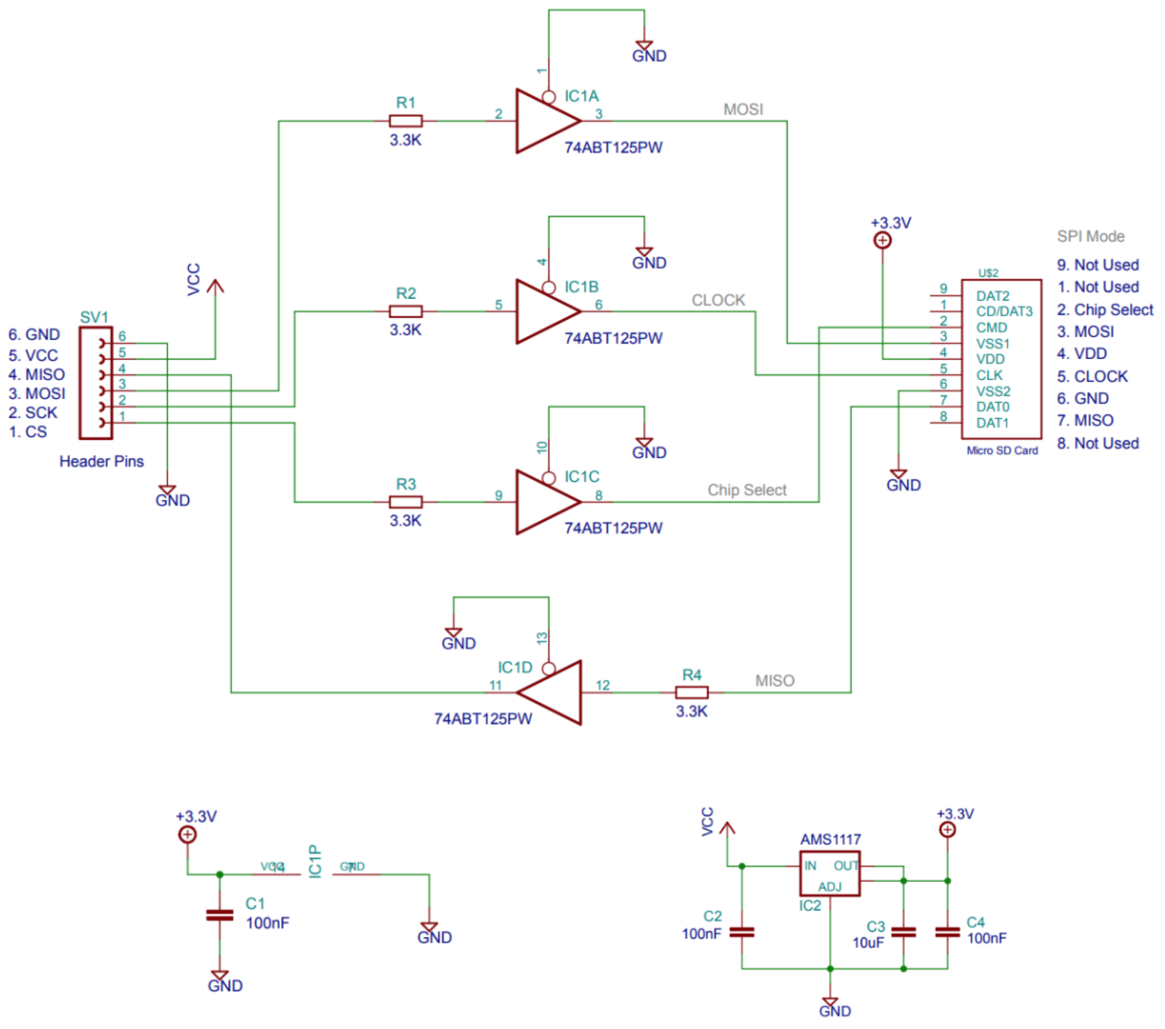

Figure 9: SD card module [7]

7.1.5 Appendix E: audio output with amplifier

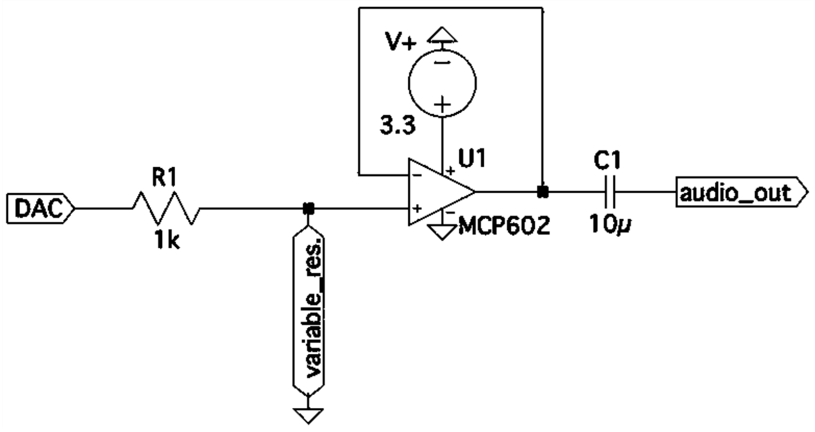

Figure 10: Audio output with amplifier

7.2 Pin configurations

7.2.1 Appendix F: STM32 pin configuration

Table 3: STM32 pin configuration

Pin	Signal
E5V	Power supply (5V)
TX (CN3)	UART to TIC
PB5	Button 1
PB9	Button 2
PA6	Button 3
PB8	Button “rec”
PB4	Button “stop”
PA7	LED 1
PB6	LED 2
PC7	LED 3
PA9	LED “rec”
PC3	Microphone AOUT
PB10	SD CLK
PC2	SD MISO
PC1	SD MOSI
PA4	Audio output

7.2.2 Appendix G: TIC pin configuration

Table 4: TIC pin configuration

P13 pin	Signal
1	9V battery
2	9V battery
3	Button “stop”
5	Button 1
6	UART from STM32
7	Button 2
9	Button 3
11	Button “rec”
12	5V power
13	Headphone audio output
14	3.3 power
15	GND
16	GND

References

[1] Barnard, A and Visagie, L., 2020, *EDesign 2020 Project Definition Document*, v0.5, SunLearn, last accessed 5 June 2020, <https://learn.sun.ac.za/course/view.php?id=54305>

[2] *LM78XX, LM78XXA - 3-Terminal 1 A Positive Voltage Regulator*, 1st ed. Fairchild Semiconductor Coorporation, 2014, p. 18.

[3] *MCP1700 Low Quiescent Current LDO Data Sheet*, 5th ed. Arizona: Microchip Technology Inc., 2018, p. 14.

[4] *LM386 Low Voltage Audio Power Amplifier Datasheet*, 3rd ed. Dallas, Texas: Texas Intruments, 2017.

[5] L. Visagie, *SD Card Low-Level Interfacing*. 2020.

[6] Waveshare, *Sound Sensor Schematic*. 2015.

[7] V. Aiea, *CATALEX Micro SD Card Module*. 2017.