OmniDrum, Group 41 Senior Design Project

Wylde Simpson, Jameson Palmer, Soufiane Louiba

***Abstract* — This paper covers the overall design and implementation of the OmniDrum system. The system contains many different drum elements which will communicate to a central controller through the use of force sensors and ADC pins. The central controller will then mix and play sounds out of an internal speaker or through an auxiliary output jack. The system will be modular in nature and will allow the user to connect and disconnect desired drum components at will. The system will also output MIDI information to an external device provided by the user if desired**

***Index Terms* — Audio, Electrical Engineering, STM32, Drum, Electric Drum Kit, MIDI**

INTRODUCTION

Electric drum kits have been commercially available since the 80's and are chosen by many musicians for a number of reasons. Electric drum kits are composed of various pads that imitate conventional drums in shape and size. These pads are sensitive to being struck with a drumstick and will send a signal to a central unit that plays a drum sound according to which component is being struck. These pads can also be sensitive to the velocity of the strike and change the dynamics of the sound to match the playing style of the user.

Musicians choose electric drum kits for a variety of reasons. Electric drum kits are much quieter than conventional drums which is useful during practice in settings such as apartment buildings or other locations where creating loud sounds may not be acceptable. Another reason someone might choose an electric drum kit is because of their versatility. The sounds produced by the kit can be changed and usually electric drum kits will come with a variety of sounds built in which allows the user to experience the sounds of many different types of drums without having to own each one individually.

Conventional drum kits can consist of instruments that wear with use or can have serious damage if misused, mishandled, or damaged in transport. These instruments can combine to a significant financial cost. This cost can be an increased liability should anything get damaged. Digitizing the sounds each instrument makes not only allows someone to play these recordings back, they can be preserved through years and decades with no worry of sound degradation unlike their acoustic counterparts. Cost of replacing damaged components with this electric drum kit is reduced by the components used and modularity in the design.

Electric drum kits are also useful for recording and composition purposes. Generally with a conventional drum kit, recording someone playing the drums can be complicated and expensive. It may require buying multiple high quality microphones, mixing equipment, and even acoustically treating the room being used. The costs here quickly add up and may be above the budget of many players. Electric drum kits can be plugged directly into recording equipment without the need for complicated setup which reduces the entry cost to recording drums. Overall there are many reasons that musicians may want to purchase an electric drum kit as opposed to a conventional one.

FUNCTIONALITY

Users of OmniDrum will be able to attach a number of sensors to different surfaces of their liking which will correspond to a certain component of the drum set. These sensors will detect vibrations and send a signal to a central hub which will include a power supply board, microcontroller for processing information, speaker for standalone practice, amplifier board for the speaker, an optional headphones port, and a MIDI Out port.

The central hub will connect to each of the sensors and play sounds out of the speaker corresponding to which sensors receive input. The sensors may also have the function to detect the magnitude of the vibrations and thus the velocity of the strike. This will allow the hub to adjust the volume and sound according to how hard the surface was struck. The user will also be able to choose between a variety of built-in libraries with different styles of drum kits to adjust the sounds to their liking. By default the drum sounds will be played out of an onboard speaker with a knob for volume adjustment. This allows the device to be used as a standalone drum kit allowing the user to practice without the need for a separate computer. There will also be a jack to connect a pair of headphones to the device in case the user needs to be silent or prefers to listen through their own device. When the headphone jack is being used, the onboard speaker will be muted. The headphone jack can also be used to connect the OmniDrum to an audio interface and allow its sound to be recorded.

Another main function of the device will be its ability to output MIDI information based on the player's inputs. MIDI (Musical Instrument Digital Interface) is a ubiquitous communication standard used to allow musical instruments to communicate with computers and other devices. MIDI allows these devices to transmit what notes are being pressed, the velocity of the notes, and more information regarding

adjustments made to parameters. Most common DAWs today are heavily reliant on the MIDI standard for creating a seamless bridge between hardware and software. Certain MIDI note values are commonly used to represent different drum hits and sounds. The OmniDrum will be able to communicate what drums are being hit and with what velocity to a DAW allowing the MIDI information to be recorded. It should be noted that while MIDI transmits note values and velocity, it does not transmit audio. This allows the user to choose any drum sound they have on their computer and assign it to each MIDI note being transmitted from the OmniDrum, making it an even more versatile product. The MIDI capability allows the OmniDrum to be used in the workflow of an amateur musician to produce complex and interesting drum rhythms that can be directly incorporated into their music. This is also especially helpful for traditional drummers to get their ideas out of their heads into their projects in a more familiar and natural way, as opposed to clicking around on a sequencer with limited capabilities.

GOALS AND OBJECTIVES

Basic Goals:

The device will be able to detect input from sensors and play a corresponding sound

The device should be able to play at least 3 sounds at once

The device should be able to output MIDI signals corresponding to which sensors are triggered

Sensors should be able to be placed on most hard surfaces and detect a hit

Advanced Goals:

The device should be able to detect the velocity of each hit and play a different sound depending on the velocity

Sensitivity of the sensors should be adjustable through a potentiometer

The device should have a headphones/line out jack, which mutes the onboard speaker when in use

Integrate an amplifier for headphone usage listed above. The amplifier may be disabled as to not interfere when connected to another sound system to not add unnecessary gain.

The device should have multiple sets of built-in sounds

Stretch Goals:

Stylization such as waveform or frequency spectrum display

Modularity such that an instrument may be added/removed with ease.

Bump/crosstalk detection to prevent sensors from picking up vibrations on the frame from accidental knocks or another struck sensor.

Demo mode where sample drum routines could play on their own while lighting up the instrument that is played. A user may follow along the routine by striking an instrument when it flashes.

A DESCRIPTION OF MIDI

MIDI (or Musical Instrument Digital Interface) is a music communication standard that has been around since 1983. To this day it is widely used and it can be found almost everywhere that you would need to digitally transmit note values and velocity values as well as numerous other functionalities. MIDI is an extremely capable communication standard that is also extremely simple to implement.

Messages in MIDI are sent over UART at a baud rate of 31250 Hz. The messages consist of 8-bit segments transmitted over the UART line to convey different properties about the note including when it begins, when it ends, the velocity of the note, what MIDI channel it is being transmitted on, and other variables such as MIDI Control Change messages that may be controlled by different knobs or faders on the MIDI controller being used by the user to change the characteristics of the sound. For our purposes, we are concerned with how to start a MIDI note, end the MIDI note, define the correct note value, and define the velocity with which the MIDI note should be played.
MIDI messages consist of different bytes which serve different purposes. The first byte in a MIDI message is the Status byte. This byte defines the type of message being transmitted. Following the Status byte are Data bytes which convey further information about the specific message defined by the Status byte. Once a Status byte is sent, the system will assume all subsequent Data bytes will be in the context of the most recent Status byte received. This means that if you are transmitting only Note messages you do not need to resend the Status byte for each note.

HARDWARE DESIGN

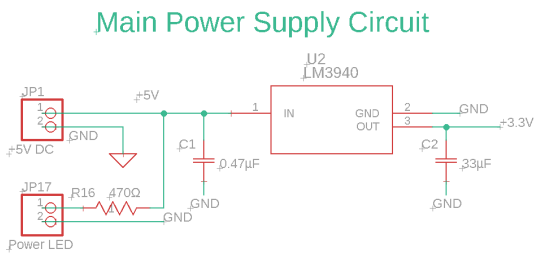


Fig 1. Main Power Supply Circuit

Shown here is the schematic for the Main Power Supply Circuit. The power will be provided at JP1 from a 5V AC/DC power supply plugged into a barrel jack which will be mounted on the outside panel of the unit. This will form our +5V line and our GND. The +5V line then goes down through a 470Ω resistor out to JP17. JP17 will power an LED that will be mounted near the DC barrel jack to indicate that the device is being powered. The +5V line will then go into the LM3940 LDO regulator which will output a voltage of 3.3V to power the MCU and other components. The input line of the LM3940 has a 0.47µF capacitor connected to it and ground as was recommended in the datasheet, there is also a 33µF capacitor on the output connected to ground. The 33µF capacitor is electrolytic with the negative side connected to ground, it was also recommended to have an ESR rating within a specific range, the ESR of this capacitor is 420mΩ.

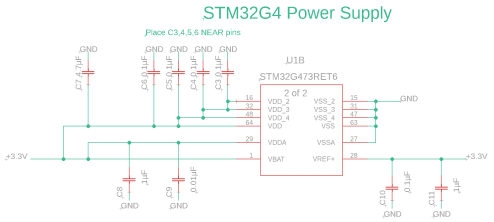


Fig 2. STM32G4 Power Supply Circuit

This circuit shows the power supply scheme for the STM32G473RET6 microcontroller. The schematic split the microcontroller into two separate parts: one for the power supply pins, and the other for the IO pins. Above is the part for the power supply pins. The input voltage for the microcontroller is +3.3V. Several capacitors are shown connected to ground around many of the pins; this practice was recommended in the datasheet for the microcontroller in order to improve stability. It should be noted that capacitors C3,4,5,6 are explicitly recommended to be placed as close to their respective pins as possible for the best results. Due to this, the components shown here were specifically chosen to be SMD/SMT components. The rest of the components on the board are through-hole components due to simplicity of removing and replacing through-hole components when testing and fabricating. Through-hole components however would not have been able to fit as closely to the pins as desired. The capacitors here are all of the 1206 package size.

The VDD pins provide power to the main internals of the STM32G473RET6 while that VDDA pin provides power specifically to the analog portions of the microcontroller. The VDDA pins were given a different series of capacitors for stabilization, but receive voltage from the same +3.3V line. The VREF+ provides an external voltage reference for the analog circuits. There is an internal voltage reference for the analog circuits as well, however it is best practice to power this one. The VBAT pin was recommended to be given direct +3.3V without any capacitors required to be near the pins. The VSS and VSSA pins are all connected directly to ground.

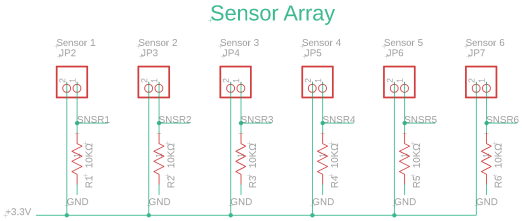


Fig 3. Sensor Array Circuit

The sensor array consists of the six sensor circuits shown above. The schematic shows JP2 - 7, acting as pin headers for the individual sensors. These pins will be connected to a series of panel mounted 3.5mm TS jacks. The jacks feature two conductors and will allow us to easily connect and disconnect individual sensors to our main board. These force sensors will provide us with measurements of how hard each component of the drum kit is being hit. The leads of the sensors will be connected to a 3.5mm TS cable.

The force sensors act as a variable resistor with two leads. We will connect one of these leads to an ADC pin on our microcontroller; each sensor will connect to its own individual ADC pin. According to the STM32G4 datasheet, the 12-bit ADCs read voltage values from 0V to +3.3V and output a digital value from 0 to 4095. The ADCs cannot sense a change in resistance, so we will need to make a voltage divider to enable the functionality of the sensors. To do this we will place a 10KΩ resistor in parallel with the ADC pin that will be connected to ground. When unpressed the sensor should be around 20MΩ. If we then apply +3.3V to the other lead of the sensor, we can expect the voltage at the ADC pin to be about 1.6mV. This should be seen as a near zero digital output value from the ADC pin. Conversely, when the sensor is pressed fully it should be near 0Ω in resistance. In this case the voltage seen by the ADC pin would be +3.3V and should register as a 4095 digital output value. This setup allows us to utilize the full range of the sensor as digital values from our ADC.

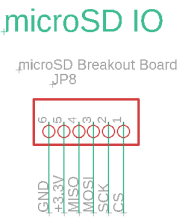


Fig 4. microSD IO Circuit

The microSD IO schematic shown above is very simple. The 6 pin header will connect to a breakout board that contains a microSD port so that it may interface with the microcontroller. The GND and +3.3V will be supplied by the main board and the other 4 pins will connect and communicate with the microcontroller over SPI.

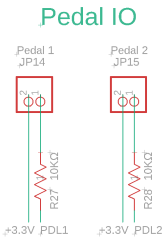


Fig 5. Pedal IO Circuit

The Pedal IO shown above functions similarly to the profile select button as a digital input. The pedals should be connected to the +3.3V line and a GPIO pin set as a digital input with a pull-down resistor. When a pedal is unpressed the digital input pin that it is connected to will read as low. When the pedal is depressed it will bridge the pins together and the input pin will read as high. The pedals will be connected using panel mounted ¼" TS jacks. Similarly to the profile select button, a 10KΩ resistor has been added in line with the pedal output as protection.

The two pedals will have different purposes. The first pedal will be used to simulate a kick pedal on a drum kit. When pressed down, the system should use the pedal as an input trigger to play a kick drum sound in the same way that the force sensors are used. The second pedal will be used to simulate the pedal on a hi-hat cymbal. This pedal will not trigger any sounds itself, but will be used as a check when the hi-hat force sensor is struck. If the hi-hat pedal is unpressed when the hi-hat force sensor is struck, the system should play an open hi-hat sound. If the hi-hat pedal is pressed when the hi-hat force sensor is struck, the system should play a closed hi-hat sound. It should be noted that usually the act of pressing the hi-hat pedal is used to make its own sound, however as the pedal is a digital input in our case there is no way to know how loud the user would like the sound to be and the sound may get in the way of other sounds when playing if it is too loud.

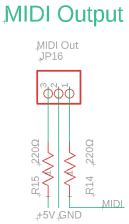


Fig 6. MIDI Output

Shown above is the schematic for the simple MIDI output that will be incorporated in our project. The 3 pin header will be connected to a female 5-pin DIN port on the side of the assembly. This port will connect to an external MIDI capable device to transmit the notes being played or to be recorded in a user's DAW. The MIDI output is powered by the +5V line and contains two 220Ω resistors. By design a 220Ω resistor should be placed across the +5V line to limit current. A 220Ω resistor is also placed on the UART transmitting line in order to protect it as well. These pins will connect to the MIDI output port either through connectors and pin headers, or wires directly soldered into the board.

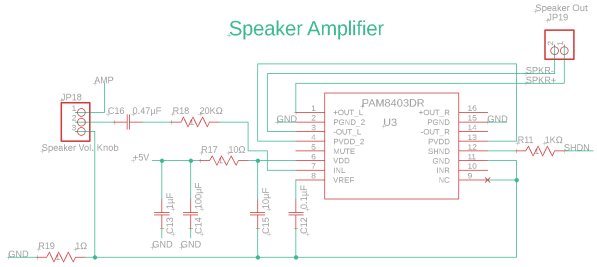


Fig 7. Speaker Amplifier Circuit

The speaker amplifier shown above uses the PAM8403 to drive the internal speaker by amplifying the sound coming out of the DAC pins of the microcontroller. The amplifier features many peripheral components to increase stability and to ensure the proper function of the amplifier.

The audio signal initially arrives from the 'AMP' line shown at the top left of the schematic. This line is connected to all of the DAC pins (Pins 18, 19, 20) on our microcontroller. The DAC pins on the microcontroller are buffered which means that they are protected from extraneous current and can be directly connected to the input of the amplifiers. This also allows us to tie the three pins together to add the signals so that all of the audio signals go to the same destination regardless of the DAC that is outputting. This also allows the use of multiple DACs simultaneously – if needed – to drive the amplifiers.

In this schematic, the AMP pin is connected to one of the leads of a potentiometer. This appears as a 1x3 pin header in the schematic because the potentiometer will

be mounted to the outside of the device assembly. The potentiometer will connect to the main PCB at this point either through the use of a removable connector, or by soldering wires directly onto the board. This potentiometer acts as a volume knob for the speaker amplifier, allowing the user to adjust the volume to a desired level. The value of this potentiometer is 10KΩ; this was selected because it was the value shown on the example circuit diagram used as a reference for this amplifier circuit.

The 'output' of the potentiometer is connected to the main input of the amplifier. With this setup, the user is able to control how much of the input to the amplifier is influenced by the audio input source, and how much is influenced by ground.

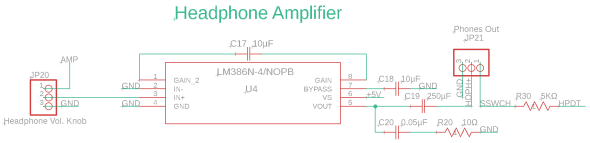


Fig 8. Headphone Amplifier Circuit

The headphone amplifier is the other component of the audio output system. This amplifier functions differently than the speaker amplifier as it drives a much higher impedance device. The assumed impedance of a device being plugged into the auxiliary output is 32Ω.

The signal flow for this portion of the circuit begins with the 'AMP' node on the left side of the circuit. This connects to the JP20 pin header; this 1x3 pin header will connect to a potentiometer that will be panel mounted to the outside of the assembly. This potentiometer will be used as a volume control knob for the user to adjust the output volume of the headphone amp to a desirable level.

The audio input from the potentiometer connects to the IN+ pin on the LM386 and gets amplified with a gain determined by the gain pins. The pins GAIN and GAIN_2 on the chip are biased using a capacitor to decide the value of the gain that is applied to the target signal. When not connected, the gain value of the LM386 is set to 20 by default. This is the lowest value and is the default in order to protect the user from accidentally damaging a device connected to the output in the case that they ignore the GAIN pins.

The audio output of the amplifier comes out of the VOUT pin. This pin is then connected to a capacitor going towards the audio output jack, and a capacitor and resistor in series going to ground. The function of the resistor and capacitor in series here is to provide a load at higher frequencies. This stabilizes the amplifier and prevents oscillations from occurring. If removed, the oscillations could damage the amplifier. Although the headphones output here is a mono ¼" TS jack, this connector has three pins instead of two. This is to facilitate the muting of the internal speaker when something is plugged into the Phones output jack. The Phones output jack is a switched jack that allows us to detect when something has been plugged into it.

A switched mono jack has four pins, two of them are used as connections to handle the conventional use of the jack. The other two pins are normally-closed switches that connect to the other two pins respectively when nothing is plugged into the jack, and open when something is plugged in. This allows detection of the jack being used without the need to rely on an external circuit. Since the switches are actuated physically they will open when any device with a ¼" connector is plugged in.

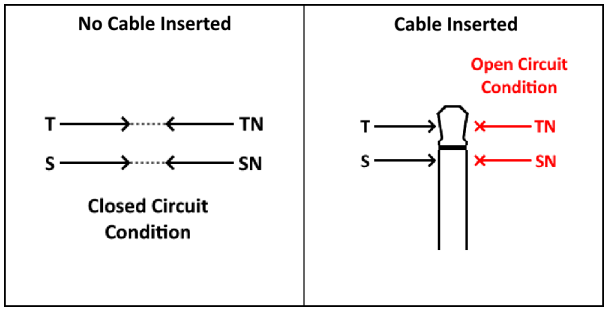


Fig 9. Switched TS Jack Diagram

The opening of the switches in the switched jack can be used to control the state of the SHND pin on the speaker amplifier. In order to shut down the speaker amplifier we need to drive the SHND pin low. To do this we must ground the pin in some way. Connecting the pin to the sleeve switch pin (SN in Figure 6.10) would allow us to ground the SHND pin, but only when nothing is plugged into the Phones jack. When something is plugged into the Phones jack the sleeve switch would open and the SHND pin would no longer be grounded. The internal pull-up resistor on the SHND pin would then enable the amplifier again. This would mute the speaker until headphones are plugged in, making it so that the headphones and internal speaker only play at the same time. This functionality would be the opposite of what is intended.

In order to make this functionality work we must first connect the sleeve switch pin of the Phones jack to a GPIO pin on the microcontroller. For this we will use Pin 42. Pin 42 will then be set as an input with a pull-up resistor. This way when nothing is plugged into the Phones jack the normally-closed switch on the jack will connect the sleeve switch to ground. This is because the negative terminal of the headphone audio pair of wires is already tied to ground.

Now, when something is plugged into the Phones jack, the switch will open and the input pin will no longer be grounded. Then the internal pull-up resistor

will pull the input pin high. This means that in the software, something being plugged into the Phones jack will appear as Pin 42 going high.

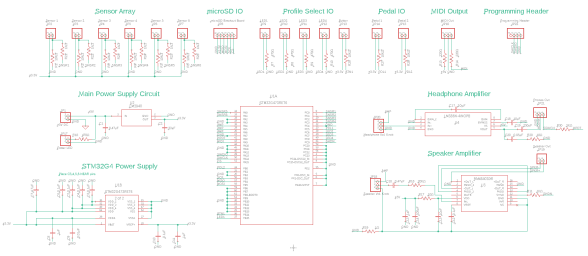


Fig 10. Overall Circuit Diagram

All together these components together create the overall schematic for our project. This is what will be included on the main PCB. The components will then need to be placed on the board according to their mounting style. The center of the image shows the IO pins of the microcontroller. Though it appears as if nothing is connected, the pins wires coming off them with the same label as many of the inputs and outputs on other parts of the circuit. In this way they are connected within the software but are not visually displayed in this way to reduce clutter. All unused GPIO pins are shorted directly to ground in order to reduce any issues that may arise from leaving the pins floating.

All of the pin headers shown in the circuit schematic will connect to some part of the general user IO. This includes indicators such as LEDs, the profile select button, the headphone jack, the sensor jack array. All of these things will be panel mounted and not attached to the board. This gives us more options on how to specifically customize the layout of our user interface so that it flows well and is easy to use.

STRUCTURE & EXTERNAL IO

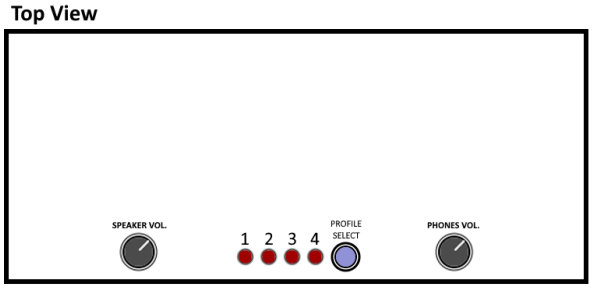


Fig 11. Central Hub Top View

Above is a rough graphical representation of what the central processing hub will look like from above. It will take on the form of a rectangular box with various IO on the different faces of it. In the image above, the bottom of the box is the side that will be facing the user when in position. On the side near the user we can see four different LEDs that will indicate which profile is currently selected. As discussed in the earlier sections, only one of these LEDs will be illuminated at a time in an effort to not pull too much current from the microcontroller. These LEDs are on a circuit that will limit their current draw to less than 10mA. These LEDs will have visible labels above them to indicate which profile number they represent. Near them will be a PROFILE SELECT button that will allow the user to round-robin through the different profiles by pressing the button momentarily and then releasing. This will change which profile LED is illuminated as well.

On opposing sides of the profile selection interface will be two knobs, one for individually controlling either the volume of the internal speaker or the volume of the headphones. This allows the user to decide the volume for the two outputs separately so that they can adjust the volume independently to their liking. Both knobs will be clearly labeled so that the user understands which knob controls what, and they will be placed in a spot that they are not in the way of the user and are not easy to bump around and accidentally adjust unintentionally so that the user doesn't accidentally hurt their ears.

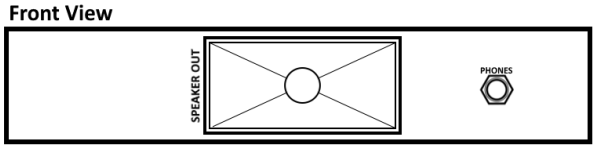


Fig 12. Central Hub Front View

This image depicts the view from the front face of the device. In this case, the front face refers to the face that is pointing towards the user, and is what they would see in the front of the device looking head on. This face has only a few features including the speaker and the PHONES port.

The speaker is depicted by the rectangular shape in the center and is a rough estimation of what our internal speaker should look like. It is possible that the final design may involve a grill or protective covering for the internal speaker that protects it from being damaged by the user or any other foreign outside object that may puncture, damage, or dislodge it. This protective grill would still allow the sound from the speaker to be heard and escape the enclosure with minimal effect to the overall quality of the sound.

The final icon that we can see on this face of the structure is the PHONES port. This port is a mono ¼" TS jack where the user can plug in any external headphones that they would like to use when practicing or recording with the device, possibly in a place where they would need to be quiet. Plugging something into

this port will mute the internal speaker and only output audio through the PHONES port. The PHONES port should be placed in a location that is both easily accessible to the user and will not cause the cable for their headphones to get in the way of their movement, or get bent in an undesirable way. It should also be in a location near the PHONES VOL. knob so that the user can easily understand which knob will control its volume.

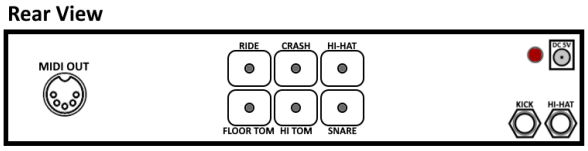


Fig 13. Central Hub Rear View

This image depicts the rear view of the device, more specifically the face of the device that will be facing away from the user when they are using it. The majority of the IO is located here including the DC Power Input, the ¼" TS jacks for the Kick and Hi-Hat foot pedals, the array of 3.5mm jacks that will connect the sensors to the microcontroller, and the MIDI OUT port that will be used to send MIDI information to an external device.

In the bottom corner (this would be the bottom left corner if viewed from the front face) are two ¼" TS jacks that are to be used for the Kick and Hi-Hat foot pedals. These are placed on what would be the left side to the user because the Hi-Hat is usually placed on the left within a conventional right-handed drum kit. The Kick port was also placed near here for convenience and to make sense with the design, with the Hi-Hat placed outside as it would be further left. These ports will be used to both tell the system when to play a kick drum sound, and when to play an open or closed hi-hat sound file. When pressed down the connection is bridged and the microcontroller will see a high digital logic signal.

In the center of the rear face is the array of 3.5mm TS jacks that connect to the different components of the drum kit. Six 3.5mm TS cables will connect to the jacks and will route power to the sensors on all of the drum and cymbal enclosures. Each port will be labeled based on which component of the drum kit it connects to. Lastly, on the left side of the rear (the right side from the user's reference point) is the MIDI OUT jack. This female 5-pin DIN jack will be used to connect a MIDI cable to an external 3rd-party capable of receiving MIDI information. This includes an audio interface that connects to a computer so that the MIDI information from the drum kit can be recorded in the user's DAW and used within their own music production process.

SOFTWARE DESIGN

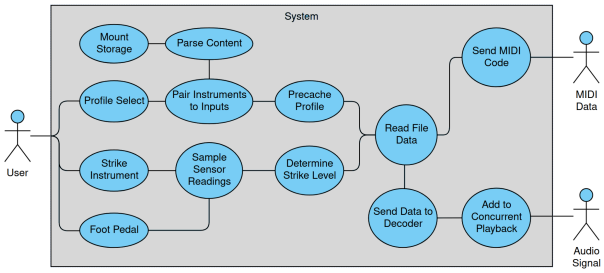


Fig 14. Software Block Diagram

A user interfaces with several modules of the device while several monitoring tasks are performed. A user action can be captured to registers where the task may iterate over these values. The captured value is used in branching behavior of the software to continue along the software flow. When the reading is passed to further process the task may continue monitoring the rest of the register values. Separating the monitoring tasks from these added processing steps is meant to increase parallelization giving greater responsiveness to our array of sensors. User interaction is the driving force for changing the state of the software at any given time. After initializing the state of the device a user can interact with the instrument modules or switch profiles. Upon switching profiles the state must reinitialize with the content associated with the given profile.

Cycling through the profiles available using the profile select button can activate the precaching scheme several times which can introduce high i/o usage and delay. The precaching scheme is part of the initialization steps whenever the state is changed. The last instance of the profile switch takes precedence over other instances of the cache to ensure maximum utilization of memory resources close to the cores. The last iteration of precache should evict most or all of the contents of other profile sound files to contain large portions or the entirety of the active profile content. It is expected for a user to not immediately need the response of the instruments when the profile switches. Transferring data from the external storage is a significantly slower operation. When a user is switching profiles there is enough of a time window to perform access to external storage for the precache scheme. We expect the first usage of profile resources to be a cache miss and get transfers to external storage so a user would normally experience part of this delay regardless. Doing these i/o transfers as a batch will lead to fewer cache misses following the initialization.

The microcontroller cache policy should be set to most recently used caching schemes to give the greatest benefit of retaining useful data close to the cores. After precaching the microcontroller cache policy will determine contents of cache as the device is used with

more frequent requests of data having a greater lifetime in cache. Data that is residing in cache but has gone stale or is not used as much as other rows of data will be evicted and the space is used for more frequently accessed rows. The entirety of a profile's audio data may not be placed inside of cache. The limited memory space of the microcontroller must share its resources with the running software, codex, buffers, and the cache space. Instances of data transfer from external storage may be mitigated by retaining beginning segments of files while a retrieval is sent for the remainder. Audio files are arranged in sequence with playback times so the beginnings of these files can have access through the cache in the time necessary to retrieve following sequences.

Without external storage available the device will not receive a response from the storage monitoring task. In the event the storage card is not inserted or removed the device will enter a standby mode until a response is received from an external storage device. Without storage the current mounted filesystem will be unmounted and tasks which are used for instruments will be suspended while the system awaits a response. Upon reinsertion the storage communication is then reestablished by the SPI initialization steps. The filesystem is then remounted and sent through the parsing stages. In the event a storage device is inserted that does not have a supported filesystem the software will await a properly formatted device to be present. Parsing and coursing the directories will initialize the state of our system again to ensure previous mounted data is no longer valid. A fresh state of the system is created before resuming suspended tasks.

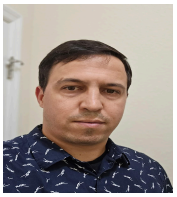
Parsing the storage will present the structure of our sound library. The number of profile folders available will be determined and the first 16 profiles that are discovered will be taken. A structure array will contain the available profiles in an object-like fashion where the current profile will cycle through the array and wrap around to the first index. Whenever the profile select button is pressed the index will increment and take the information for that index to the profile setup steps. Information in each index contains the pertinent information of how many strike levels per instrument and the associated files or MIDI codes. Assignments made from previous profiles are overwritten or set NULL where applicable. This is to ensure the system state is associated to one profile.

Within each profile the information of how many sound files get associated with an instrument is saved. This value is used to divide the range of sensor values evenly and assign the sound file for each range. Calibration of sensor readings will be done to determine a MIN and MAX strike to find what range of sensor readings gets divided. A strike value passing a given threshold determines the level of strike that is played. To ensure our thresholds are appropriately placed the MAX strike should have an offset so the sensor readings may pass this threshold rather than have a MAX strike lie on this threshold. Without this offset a strike which should be detected at a higher level may be read as a strike from a level below.

THE TEAM

Wylde Simpson
Electrical Engineering

Jameson Palmer
Computer Engineering



Soufiane Louiba
Electrical Engineer

REFERENCES

[1]"General MIDI Drum Notes." MIDI Drum Chart, zendrum.com, www.zendrum.com/resource-site/drumnotes.htm.

[2]"Lm386." Texas Instruments, https://www.ti.com/lit/ds/symlink/lm386.pdf?ts=1732638357489

[3]"Lm3940." Texas Instruments, https://www.ti.com/lit/ds/symlink/lm3940.pdf

[4]"PAM8403-247318." Diodes Incorporated, https://www.mouser.com/ds/2/115/PAM8403-247318.pdf?srsltid=AfmBOoq5JOmVQgSXmgj7009gIm42bYmWdBN9q8-EsHbHalkg35UzcRLQ

[5]"STM32L562E-DK." STMicroelectronics, www.st.com/en/evaluation-tools/stm32l562e-dk.html#st_all-features_sec-nav-tab.

[6] "The MIDI Specification." MIDI Specification, people.ece.cornell.edu/land/courses/ece4760/FinalProjects/s2000/selan/midi.html.