Optoelectronic Saxophone

Manuel Correa, Andrea Styles, Tim Walsh

Dept. of Electrical and Computer Engineering/College of Optics and Photonics, University of Central Florida, Orlando, Florida, 32816

***Abstract* — A major concern for musicians is finding a location to practice or perform that does not agitate the peace of surrounding people. Our project provides a solution to this in the form of a "silent sax," where the music plays electronically through common listening devices, such as earphones or headphones, as opposed to relying on acoustics. Unlike other electronic wind instruments on the market, this saxophone operates optically using semiconductor laser diodes and photodiodes to detect user input. A microcontroller then processes the input to assign a note to a combination of keys that match the fingerings of a saxophone.**

***Index Terms* — Music, diode lasers, photodiodes,**

I. Introduction

One of the most challenging hurdles when learning a new instrument is overcoming the anxiety of playing when others are nearby. Often people become overwhelmingly self-conscious of their playing. They may believe that the repetitive playing that comes with practicing might irritate those around them. They may also prioritize others' comfort, choosing to play softly at the cost of forgoing body or lung strengthening needed for proper playing. Even seasoned musicians are not exempt, as playing expressively may lead to noise complaints from neighbors.

This anxiety is even more prevalent in today's day and age with the COVID-19 pandemic as the virus has resulted in international lockdowns. More civilians have begun to seek tranquility at home, and the public's fear of the disease has led to an increase in irritability and nervousness [1].

The Optoelectronic Saxophone provides a solution to playing or practicing in one's home without disrupting the peace of those around them. The saxophone features an AUX connector to output audio through headphones. If playing for others, the saxophone also has a built-in speaker with volume knob to control the intensity of the sound.

Most importantly, to distinguish itself from other electronic instruments in the market, the saxophone primarily functions using a system of laser diodes and photodiodes that respond to the user's inputs, which will in turn be processed to produce the desired notes.

II. System Components

The saxophone, more formally referred to as the system, features multiple subsystems that comprise of system components. These parts are either purchased or designed, and the interconnection between them results in the construction of the final product. This section serves to familiarize the reader with the aforementioned components on a functional and somewhat technical level.

A. Laser Subsystem

The laser subsystem consists of eight pairs of laser diodes and photodiodes. The laser diodes act as transmitters and the photodiodes as receivers.

1) Laser diodes: The laser diodes used were the OPV310 vertical cavity surface emitting laser manufactured by Optek / TT Electronics [2]. It outputs 1.5 milliwatts of optical power at a 850 nm peak wavelength, putting it in the near-infrared region. The diode begins to lase at a threshold current of 3 milliamps, and its operating forward voltage is 2.2 V.

2) Photodiode: The photodiodes utilized in the subsystem were the INL-5APD80 photodiodes produced by Inolux [3]. This photodiode has a P-I-N structure as opposed to the simpler P-N configuration. Its spectral bandwidth ranges from 400 nm to 1100 nm, though it has a peak sensitivity at 850 nm. With no illumination, it can produce a dark current as small as 5 nA. It has an open-circuit voltage of 0.4 V and a short-circuit current of 35 μA. It also offers a fast rise and fall time at 45 μs each.

3) Keys: The keys were 3D-printed using 1.75 mm black PLA 3D filament from the company Inland. The spools used come at a total weight of 1 kilogram.

a) Springs: The keys also used 304 stainless steel springs from the company Uxcell. They have an 11 mm outer diameter with a wire size of 0.8 mm. Uncompressed they measured 20 millimeters in length, and they compress down to 12 mm. These springs have a load capacity of 11 N and a spring rate of 1.38 N/mm.

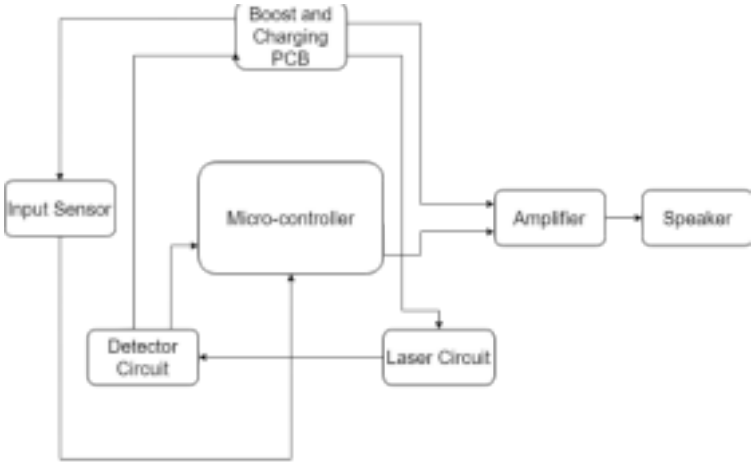


Fig. 1 Flowchart of complete system showing the connections between subsystems. .

B. Microcontroller

The microcontroller is an ATMega 2560 with purchased by Arduino. As such, the project uses the Arduino integrated development environment. It has 54 digital input/output pins and 16 analog inputs as well as 4 hardware serial ports, a USB connection, a power jack, an ICSP header, and a reset button.

C. Battery

The batteries used were two of the 3.7 V 200 mAH 18650 batteries.

D. Peripherals

1) Mouthpiece sensor: For the mouthpiece, we elected to use a CNY70 proximity sensor.

2) Audio: The speaker used for the saxophone was an 8 Ω magnetic speaker with an adjustable potentiometer to acts as a volume knob.

E. Casing

Just like the keys in the laser subsystem, most of the case was 3D-printed using Inland's black PLA. Both epoxy resin and pairs of 4-millimeter hex nuts and machine screws were used to join the surfaces together.

All materials used are RoHS compliant – that is, they do not have or exceed the maximum permissible quantity of hazardous materials, as dictated by the Restriction of Hazardous Substances directive imposed by the European Union [4].

III. System Detail

This section aims to describe the complete system by thoroughly detailing the operation of the subsystems as well as the interfacing between the subsystems. Fig. 1 above illustrates a flow diagram of the optoelectronic saxophone.

A. Power system

Since the switch to using an Arduino as our main processor due to the unexpected withdrawal of our fourth team member, there was a bit more freedom in terms of powering our device which requires 5V to operate. The Arduino has an integrated 5V regulator and with this it can be powered with a 1000mAH 9V battery or with a 3.7V 2000mAH 18650 battery. This device is powered with 18650s using a board with a TPS6109 and a MCP73871. The TPS is a 5V boost regulator used to boost the 3.7V of the batteries to 5V with 92% efficiency. The MCP is an IC that allows for passthrough charging of the batteries, which means the device can be used and powered via micro USB while the batteries are charging. This approach was chosen since using 9V isn't as common anymore in devices to the end-user but would

have worked just as well as the 18650's with the same device usage. The 18650s chosen are rated at 3500mAH and the device draws between 500mA-900mA depending on usage (500mA idle not in sleep mode and 900mA at peak current draw); therefore, the device has about a 3–5-hour battery life at max charge.

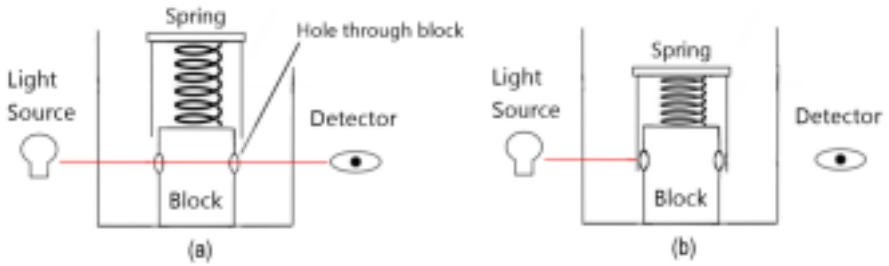


Fig. 2. Demonstration of the key mechanism. Fig. 2a shows the key at rest, allowing the light to pass through into the detector. Fig. 2b is when the key is pressed, blocking the light coming from the source.

B. Printed Circuit Boards

The Printed Circuit Boards (PCBs) used in this project came from both JLCPCB and our own at home labs. The more complex multi-layer boards were ordered from JLCPCB. The simpler PCBs, such as the CNY70 sensor board used on the mouthpiece sensor, were made by us using Ferric Chloride etching. The etching process involved taking a copper board with laminate substrate and using ink to cover the traces and pads of the PCB on the top copper layer. With the traces covered by the ink, the boards were submerged in a Ferric Chloride solution where all the copper not covered by the ink is dissolved off the substrate. The final product of this process is a laminate board with copper traces that represent the circuit. The next step of this process would be to apply a solder mask to prevent oxidation on the copper traces which would promote a longer lifespan of the board. However, out of the four consumer brands we tried for masking, none of them worked well and had an uneven cure no matter how it was applied. In future endeavors, we would likely use a commercial grade solder mask instead of the hobbyist brands available on Amazon.

B. Laser Subsystem

When the saxophone is powered on, the laser circuit is supplied with 5 V. Since the diode's operating voltage is 2.2 V, we required a resistor to hold the remaining 2.8 V. Testing conditions found in the data sheet for the OPV310 utilized a forward current of 7 mA, so to remain consistent we used the same current as opposed to the minimum threshold current of 3 mA. To calculate the nominal resistance to connect in series to the diode, we need only use Ohm's law,

$$V = IR, \quad (1)$$

and see that the necessary resistance is 400 Ω. Unfortunately, that amount of resistance was not readily available; therefore, we opted to use a 430 Ω resistance instead. Fig. 3 shows a simplified schematic for one key. For eight keys, we simply put the diode and series pair in parallel to maintain equal current.

Fig. 3. Simple schematic for a single laser diode.

One of our main goals regarding the project was to not let any light escape and reach the user or its surroundings. Our largest constraint in combatting this was the materials for the casing.

The available options were wood or 3D filament (specifically black PLA), and we elected to use the PLA as it was readily available in large quantities due to having supply in both ECE and CREOL labs at the university. The only report found on the optical properties of black PLA had inconclusive results regarding the absorption spectra of the material, quoting that it “was not transparent enough to let light pass through it [5].” This actually suited our project perfectly, as it would suggest that the light will not permeate outwards.

Next was to choose the wavelength of our laser diodes. Since we wanted to limit the discomfort of the user produced by the saxophone, we opted for near infrared laser diodes. We chose the wavelength to be 850 nm as it was the closest option to the visible light spectrum, and it satisfied our other restriction – its peak wavelength had to match the peak response wavelength of our photodiodes. If it were not for the latter constraint, we also had the option to procure a laser that was 780 nm.

A mechanism for the keys was made to simulate on and off states that would be processed by the microcontroller. A visual representation for the keys can be seen on Fig. 2. At rest, the diodes are powered and the beam travels through a hole on a cylindrical rod – which supports a spring and a tube with a closed end acting a stopper – and onto the detector. Once a key is pressed, the stopper goes down and blocks the beam.

The photodiode used was the Inolux INL-5APD80 PIN diode. To find the load resistance needed for the photodiode (1) was used with the open-circuit voltage and short-circuit current found in the datasheet, resulting in a resistance of 130 kΩ. Just like the photodiode, the photodiode and resistor were connected in series, and all eight pairs were joined in parallel. When the diode beam shines onto the detector at rest, the diode produces a current of 35 μA, and when the key is pressed down to block the light it will produce as little as 5 nA. Fig. 3 shows the schematic of the photodiode.

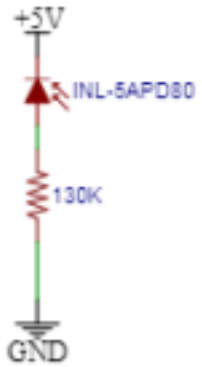


Fig. 4. Simplified schematic of the photodiode.

C. Mouthpiece sensor

In order to best simulate the effect of a traditional wind instrument where the velocity of air being blown into the mouthpiece will produce a louder sound, a microphone was used initially and the amplitude of the signal from the microphone scales the output volume. The issue with this approach was when the microphone was being blown into, the signal from the microphone would be clipped. Our next approach was taking a CNY70 proximity sensor and placing it under a membrane such as a piece of latex. The CNY70 is a relatively simple device that consists of an infrared (IR) beam and a phototransistor. As a reflective surface approaches the CNY70, the IR beam will reflect off of the surface and back into the phototransistor which will result in different output voltages.

As the user blows into the mouthpiece, the membrane expands and gets closer to the CNY70. The output of the CNY70 is a DC voltage which ranges from 5V to 3.7V and varies linearly with the membrane's proximity to the CNY70. With this, the device can accurately take the input from the user blowing into the mouthpiece and convert this into usable data for the attack, gain and decay of the output MIDI signal.

D. Audio Output

For the audio output, the TDA7052 was used which is an low power audio amplifier integrated circuit with fixed gain. The TDA7052 operates on a 5V supply signal and can drive an 8-ohm speaker at 1 Watt completely on its own with no need for any other components (although capacitors to filter any noise from the power supply and audio signal lines were used). To adjust the volume of the output, a potentiometer was placed between the audio signal input on

the board to the input pin on the TDA7052 which will adjust the amplitude of the audio signal going into the TDA7052. Therefore, adjusting the knob on the potentiometer will lower or raise the audio level.

The speaker chosen was an 8-ohm magnetic speaker that is fairly sensitive to best match the signal from the TDA7052 since the gain isn't necessarily very high compared to other audio amplifier configurations.

E. MICROCONTROLLER

In this section, the major hardware components that are used for computation and processing will be explained in detail. The Analog-to-Digital Converter is built into the Arduino Mega 2560 board, while the SparkFun Musical Instrument Shield is not.

1) Analog-to-Digital Converter: In order to convert the voltage output from each of the photodiodes into a format that can be interpreted by the software, the ATmega2560's Analog-to-Digital Converter (ADC) unit is used. The ADC utilizes successive approximation to convert an analog voltage signal to a 10-bit digital value. The minimum value corresponds to the input voltage of the GND pin, and the maximum value corresponds to the input voltage of the AREF pin, which for this project is the 5 V input to the photodiodes. Using the AREF pin disables use of the chip's internal reference voltages, as they are shorted to the external voltage. A 10-bit resolution yields a maximum digital value of 1023 and a minimum value of 0. A typical conversion takes 13 clock cycles to execute, putting the time required in the microseconds, and is performed upon each call of analogRead() within the system's code.

2) Musical Instrument Shield: With the loss of a group member during the late stages of design, the design and implementation for storing audio data and generating sound had to be simplified in order to complete the project in time. The original design involved a Raspberry Pi board connected to the Arduino Mega 2560 to act as a peripheral device. The Arduino would receive the voltage output from each photodiode, convert the voltages to digital values, then pass those values to the Raspberry Pi, which would have handled determining which note was to be played and playing the note.

The device chosen to replace the Raspberry Pi is the SparkFun Musical Instrument Shield, which attaches to the Arduino Mega through header pins. The shield runs on the VS1053b, which is a single chip audio decoder capable of producing MIDI tones. The shield acts as a system slave to the Arduino and receives commands and data through an SPI bus. The chip comes with a preloaded MIDI tone bank that includes an alto saxophone. The board also has a built-in AUX out port into which any speaker can be connected.

The shield receives MIDI signals through a serial connection on digital pin 3, instead of through the RX and TX pins. This requires a software-based serial connection instead of a traditional connection supported by the UART module, which will be discussed later. Digital pin 4 acts as the reset pin and must be written low and then high in order to reset the shield each time the system is powered on.

F. MIDI

While it does not use the traditional 5-pin connector, the instrument shield communicates through MIDI using the same specifications. The interface operates at 31250 baud, and each byte to be transferred requires 1 start bit, 8 data bits, and 1 stop bit, which results in a transfer time of about 320 microseconds for 1 byte.

MIDI operates through defined messages that consist of 1 to 3 bytes. The first of these bytes is always the status byte, which determines the type and content of the message. A status byte can be viewed as 2 4-bit nibbles, and always has its most significant bit set to 1, differentiating it from a data byte. The 3 remaining bits of the first nibble indicate the type of message being sent, and the second nibble indicates the channel to which the message is being sent. 4 bits allow for 16 channels, each being able to be assigned a unique program, or instrument, and play notes simultaneously.

A status byte with a first nibble value of F in hexadecimal or 1111 in binary indicates a system message that is sent to all channels, leaving the second nibble to indicate the type of system message. These messages handle synchronization and playback of MIDI sequences. For the purposes of this project, system messages are not used, as no pre-recorded sequences are used.

The remaining status byte values from 0x80 to 0xEF indicate voice messages, which are sent to

a specific channel. These messages handle the individual parts of the instrument on a channel, including turning a note on and off, setting pitch, and changing the instrument itself. For this project, the Note On, Note Off, Program Change, and Control Change messages are used.

1) Control Change Messages: The status byte for a Control Change message has a first nibble value of B (1011 in binary). The message consists of 2 data bytes, the first of which is the value of the controller to be changed. There are 128 unique controllers to adjust, including volume, portamento, tremolo, sound brightness, and more. The second data byte is the value to assign to the chosen controller.

2) Program Change Messages: The status byte for a Program Change message has a first nibble value of C (1100 in binary). The message consists of 1 data byte indicating the value of the program to assign to a channel. The program corresponds to an instrument within a tone bank. An 8-bit value allows for 128 different instruments to be selected. Some devices may have more than one tone bank in order to store more instruments. In this case, a control change message is used to first select the tone bank to be used.

33 Acoustic Bass	65 Soprano Sax
34 Electric Bass (finger)	66 Alto Sax
35 Electric Bass (pick)	67 Tenor Sax
36 Fretless Bass	68 Baritone Sax
37 Slap Bass 1	69 Oboe
38 Slap Bass 2	70 English Horn
39 Synth Bass 1	71 Bassoon
40 Synth Bass 2	72 Clarinet

Fig. 5. Image of part of the VS1053b's melodic tone bank. Taken from the VS1053b datasheet.

3) Note On and Note Off Messages: The status byte for a Note On message has a first nibble value of 9 (1001 in binary). The message itself has 2 data bytes, 1 for the value of the note to be played, and 1 for the velocity of the note. The status byte for a Note Off message has a first nibble value of 8 (1000 in binary), and the data byte structure is the same. The note velocity determines the intensity with which the note is to be played. The note value corresponds to a note on a traditional musical scale, with the minimum value of 0 being the lowest C note, and the maximum value of 127 being a G note 10 octaves higher. Adding 1 to the note value is the equivalent of moving up one semitone and subtracting 1 is the equivalent of moving down a semitone. Adding 12 to the note value is the equivalent of moving up an octave. This instrument will have a scale of F3 to E4, resulting in a range of values from 53 to 64.

Note	Octave										
	-1	0	1	2	3	4	5	6	7	8	9
C	0	12	24	36	48	**60**	72	84	96	108	120
C#	1	13	25	37	49	**61**	73	85	97	109	121
D	2	14	26	38	50	**62**	74	86	98	110	122
D#	3	15	27	39	51	**63**	75	87	99	111	123
E	4	16	28	40	52	**64**	76	88	100	112	124
F	5	17	29	41	53	**65**	77	89	101	113	125
F#	6	18	30	42	54	**66**	78	90	102	114	126
G	7	19	31	43	55	**67**	79	91	103	115	127
G#	8	20	32	44	56	**68**	80	92	104	116	
A	9	21	33	45	57	**69**	81	93	105	117	
A#	10	22	34	46	58	**70**	82	94	106	118	
B	11	23	35	47	59	**71**	83	95	107	119	

Fig. 6. Concert scale mapped to values. Middle C is value 60.

G. Software Detail

This section will detail the software used to drive the device, including the algorithm and the library used to support it. The code that powers the system is written in C and is 140 lines long.

1) SoftwareSerial Library: The SoftwareSerial library is included with the Arduino IDE as a means to establish a serial connection through the digital pins of the Arduino board. Normally, the board's Universal Asynchronous Receiver-Transmitter (UART) component would support a connection through pins 0 (RX) and 1 (TX). The SoftwareSerial library enables the Arduino's Atmega chip to perform other functions while a serial connection is established by having software support a serial connection through different pins by replicating the functionality of the UART. Since the instrument shield's MIDI-in pin connects to pin 3 on the Arduino, a software connection through the pin is required.

The library provides several functions to ease the process of managing a serial connection and fully replicate the functions of a normal serial connection. The SoftwareSerial() constructor allows the user to initialize a connection with selected digital pins to correspond to RX and

TX. The begin() function starts the connection at a specified baud rate, up to 115200 baud. The write() function writes data to the serial connection as raw bytes, functioning the same as a UART connection.

2) Arduino Code: The code begins with the declaration of variables for each value from the proximity sensor and the photodiodes, as well as which port each component is connected to. The variables note, newNote, and playing are also created, with playing being initialized to false. The setup function begins a SoftwareSerial connection at 31250 baud on digital pins 2 and 3. Pin 3 is the MIDI-in port for the instrument shield, and pin 2 is a dummy value, since the Arduino does not read from the shield. The shield is reset by writing digital low to the reset pin, then digital high. The MIDI operation is then initialized by using control change messages to set the master volume to the maximum value of 127, since the volume will be controlled by the user using the volume knob connected to the audio circuit, and the tone bank to 0 for the melodic bank. A program change message is used to choose instrument 66, which is the alto saxophone.

Within the main loop of the code, polling is used to determine whether the instrument is being played and the note to be played. The voltage value of the

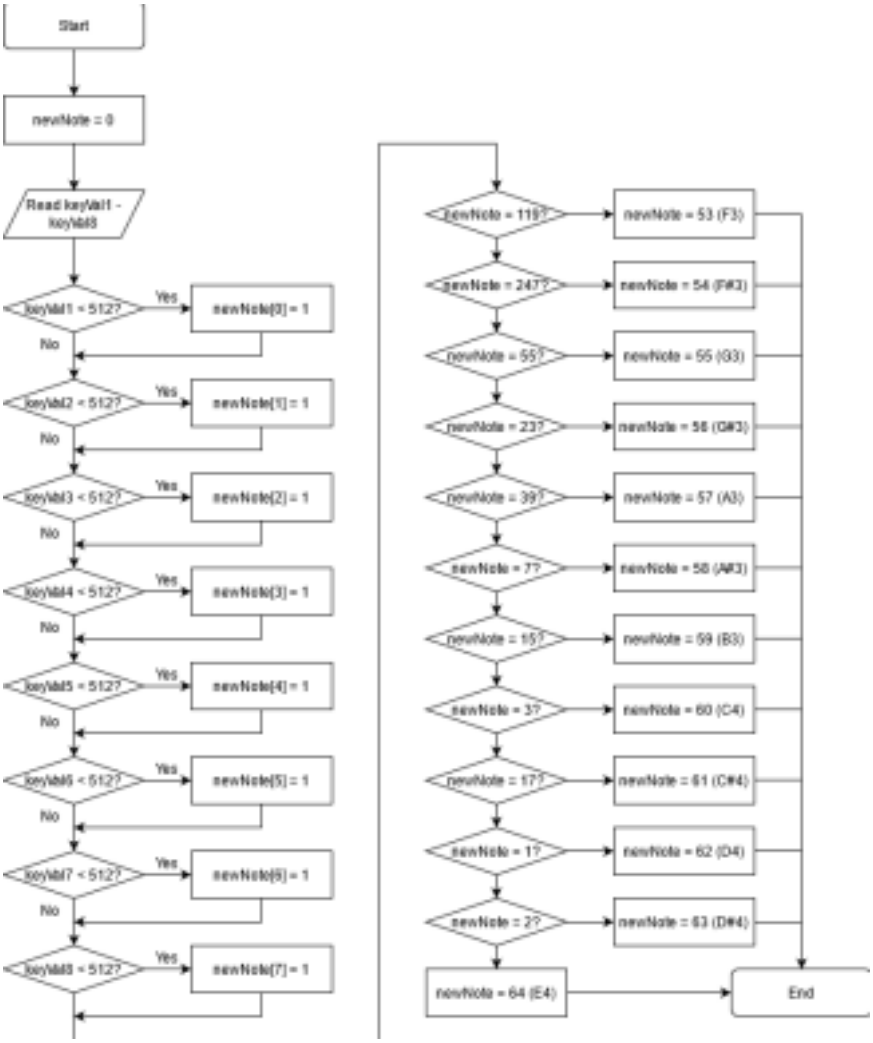


Fig.7. Note decision algorithm. Code portion executes when playing variable is set to true.

mouthpiece's proximity sensor is read into its corresponding variable, then an if-else statement is used to determine whether to set the playing variable to true or false, depending on whether the value is below the threshold of about 4.6 V,

or 950 from the ADC.

If the playing variable is set to true, then the computation of the note to be played begins. The variable newNote is set to 0, then the voltage input of each photodiode is read into its respective variable. Using each value in ascending order, (1 – 8), the bits of newNote are modified. If the value of photodiode N is less than the threshold of half its supply voltage, then bit N – 1 in newNote is set. This results in a range of distinct values that are easily mapped to notes.

A series of if-elseif-else statements is used to determine the final value of newNote. Other methods were considered, such as using an array to store the note values at the index produced by the bit manipulation, but ultimately the time difference between the methods was too small to be noticeable, so the simplest method was used. If the value of newNote is equal to a precalculated value, then newNote is set to the corresponding note value. If no keys are pressed (all photodiodes are above threshold) or a key combination is pressed that does not correspond to a traditional fingering configuration defaults to an open note, which for this instrument is E4 or MIDI note value 64. If the value of newNote is not equal to the saved value of the note variable, a noteOff message is sent to turn off the note that was previously playing, then a noteOn message is sent to begin playing the new note value. The value of note is then set to newNote to save the note value that is currently being played, then the loop begins again.

Three helper functions aid in sending messages to the instrument shield. The talkMIDI() function takes 3 parameters: cmd, data1, and data2, which are all byte values. The function sends the cmd and data1 bytes, then checks to see if the first nibble of the cmd value is A or B, which would indicate a system message or a control change message. If cmd is C or greater, then data2 is sent.

The second and third helper functions are noteOn() and noteOff(), which each take 3 parameters: channel, note, and velocity. These functions simply pass off the bytes to the talkMIDI() function, but differentiate the noteOn and noteOff commands from other MIDI communications, which are performed directly through the talkMIDI() function.

IV. Testing and Prototyping

The first point of testing was to ensure that the laser diodes and photodiodes were operating normally with no issue. When lasing the diodes operated at a current of 6.5 mA and a voltage of 2.15 V. The beam properties were not measured as the distance between the emitters and the detectors is 1.8 cm. However, a photometer was used behind the photodetector to indicate whether any light bled through the key casing or the exterior casing. Neither had light pass through the black PLA as theorized and all but one laser diode were being read by the processor – thus it was quickly replaced.

The next step involved soldering all the components together. This proved to be arduous, as the pins for the laser diode were very close together. In the end only half of the original eight keys were responsive. More effort would have been put to redo the soldering process if there was more time available. Lastly the audio circuitry was tested. Unfortunately, all instances of speakers that require soldering had failed, and we were only able to produce sound with a portable speaker.

Assembly had also failed due to complications in the designed circuit boards, which necessitated creating the circuits manually on a much larger board. Had the original PCB functioned as intended, overall assembly would be a success.

References

[1] Panchal, Kamal, Cox, Garfield (2021); “The Implications of COVID-19 for Mental Health and Substance Use”; https://www.kff.org/coronavirus covid-19/issue-brief/the-implications-of-covid-19-f or mental-health-and-substance-use/

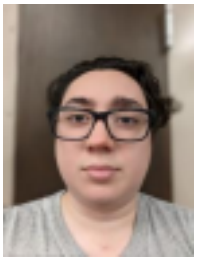
[2] TTElectronics (2018), OPV310 Datasheet, https://www.ttelectronics.com/TTElectronics/media /Pr oductFiles/Optoelectronics/Datasheets/OPV300-310Y-314Y.pdf

[3] Inolux (2019), INL-5APD80 Datasheet, http://www.inolux corp.com/datasheet/IR/Sensor/3mm_5mm_PD /INL 5APD80_V1.0.pdf

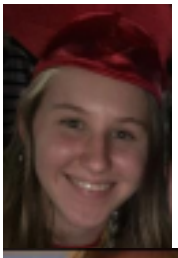
[4] European Parliament, Council of the European Union (2015); Directive 2015/863 “amending Annex II to Directive 2011/65/EU of the European Parliament and of the Council as regards the list of restricted substances”;

https://eur-lex.europa.eu/legal
content/EN/TXT/?uri=CELEX%3A32015L0863&
qid =1627413237767

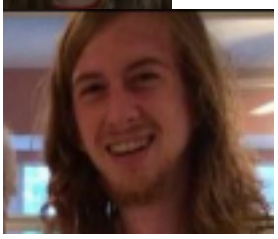
THE ENGINEERS



Manuel Correa is a graduating Optics and Photonics student who will be tackling his master's degree in Photonic Science at the University of Central Florida, specializing in optical communications and integrated photonics.



Andrea Styles is a graduating Computer Engineering student who plans to attend graduate studies in Robotics with the goal to work at Kennedy Space Center.



Tim Walsh is an upcoming Electrical Engineering graduate who will continue his studies and pursue a double major in Electrical Engineering and Computer Science.