

# CDA 6316-001 PROJECT

## INTRODUCTION AND ISSUES

This document has the purpose of illustrating the term project for the **EMBEDDED SYSTEMS AND DESIGN CLASS**.

As we all know playing a musical instrument like the piano or the keyboard requires the use of both hands, the left hand used to create bass accompaniments and the right hand used to command the leading melody or harmony. The combination of the left and right hand give sounds on two sections or ends of the frequency spectrum which sum up to give a more filled sound and complete sound.

For example if the pianist with her right hand is playing a Cmajor chord ( C<sub>6</sub> E<sub>6</sub> G<sub>6</sub> ) , with her left hand she adds some bass notes such as (C<sub>4</sub> and G<sub>4</sub>) to give the chord a better and complete feeling.



***So what if an individual with a physical impairment of lacking one arm (example the left arm) wants to play the keyboard ?***

As engineers we can offer limited solutions to these individuals where they can obtain both bass sounds and the leading melody or harmony contemporarily, But this has to be under the playing technique of just the left arm creating the bass notes to accompany whatever is played by the right hand.

We approach a solution to these problem with **MIDI KEYBOARDS**:



The midi protocol makes this problem simple to define and approach, basically **the problem becomes building a real time object or bridge that reads the notes played by the right and, interprets them and plays bass notes to accompany these notes played by the right hand therefore this object behaving as**

# CDA 6316-001 PROJECT

**bridge mimics the left arm.** This entire project proposes a solution for electronic keyboards and not mechanical pianos.

## **FLOW CHART OF THE PROPOSED SOLUTION**

So as discussed earlier, the goal is to build a bridge-like object that decides the bass note to be played based on the input notes it receives.

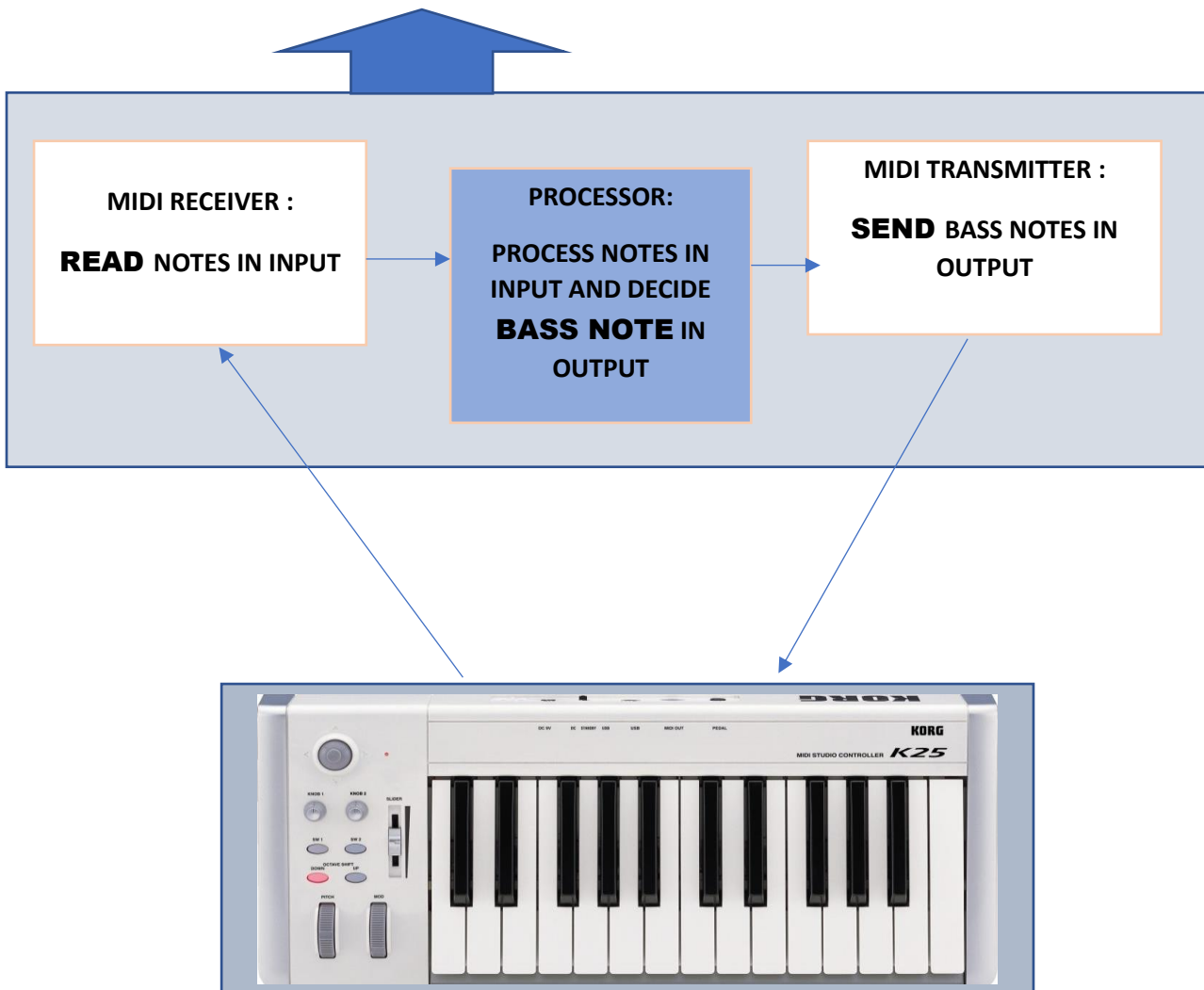
The reception of the input notes, the computation of the bass notes and the transmission of the computed bass notes have to be done in little time within the ratio of milliseconds so that the best experience is guaranteed.

The bridge-like object to be built will be called a **MED-BASSER**

And it contains 3 parts for the 3 fundamental operations to be performed.

THE **three** PARTS ARE : **1) MIDI RECEIVER 2) THE PROCESSOR and 3) THE MIDI TRANSMITTER**

## *OBJECT TO BE BUILT IN THIS PROJECT*



# CDA 6316-001 PROJECT

Components for the MED-BASSER are

1 X 6N138 Photo-Darlington Transistor OPTOISOLATOR

2 X MIDI DIN CONNECTOR

4x 220 OHM RESISTOR AND 4.7 KOHM RESISTOR

3 x LED

SOME JUMPER WIRES

A BREADBOARD

A 16 BUTTON-KEYPAD

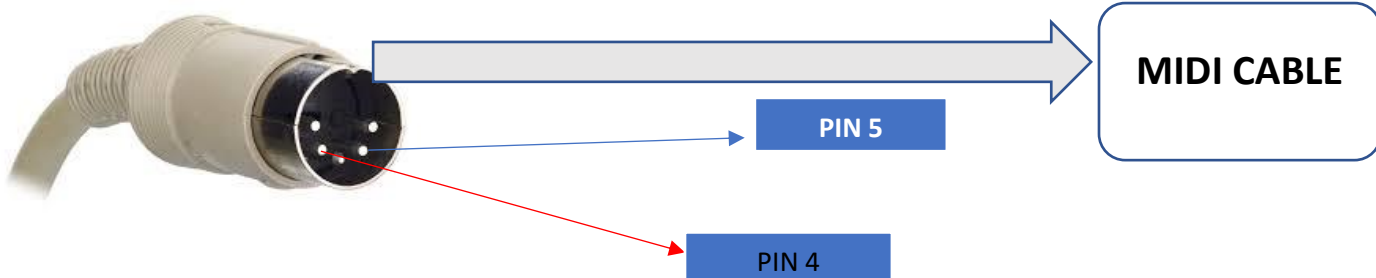
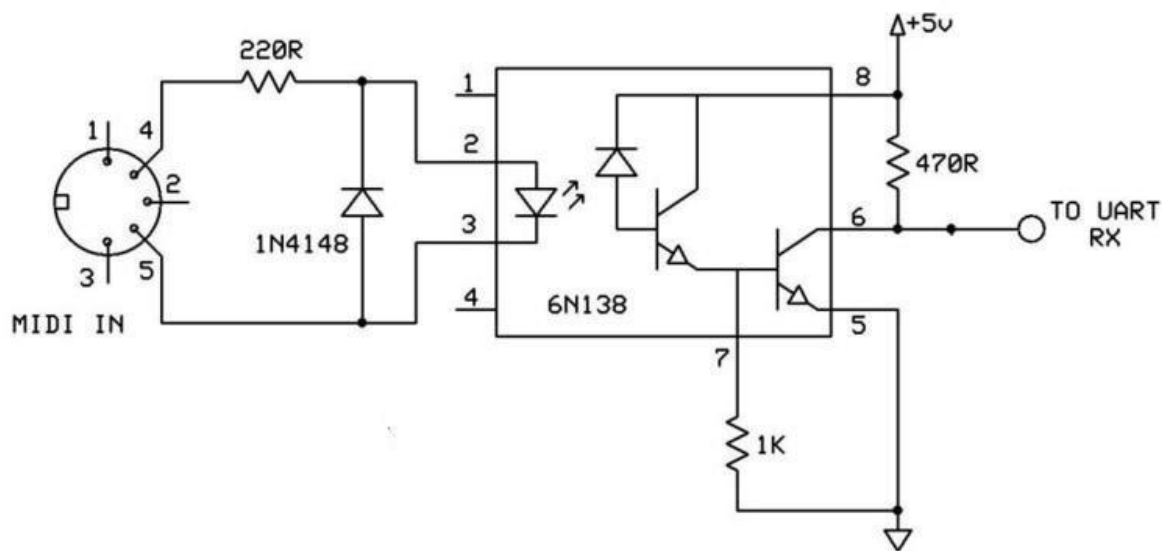
AND A MICROCONTROLLER THAT ALLOWS SERIAL COMMUNICATION

## 1) MIDI RECEIVER ARCHITECTURE:

The objective of the midi receiver is to receive the midi signal in input.

Below is the schematics for the midi receiver:

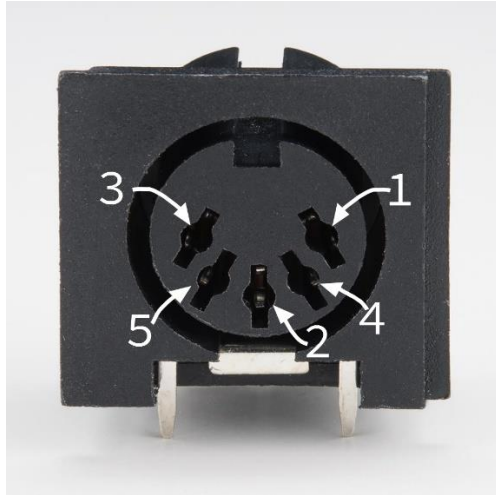
figure 1.0



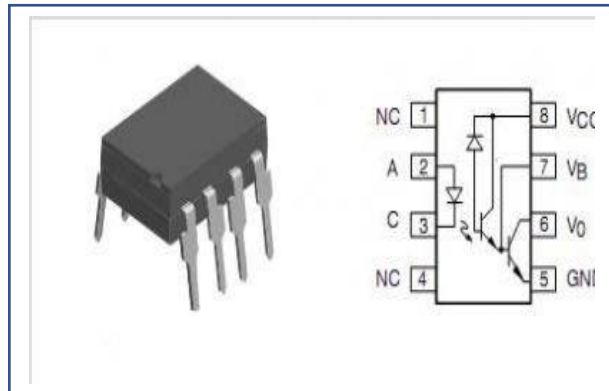
## CDA 6316-001 PROJECT

Above is a male midi connector or simply a midi cable, pin 4 and pin 5 are the pins responsible for transmitting the midi signal. For the midi receiver, a **midi female connector** will be used to receive the midi signal from the midi keyboard via the midi cable.

### MIDI FEMALE CONNECTOR



### 6N138- OPTOCOUPLER



As mentioned earlier the female midi connector receives the midi signal from the keyboard particularly in pins 4 and 5, but before the midi signal is transmitted to the microcontroller, it goes through an optoisolator or optocoupler, The optocoupler's task is to assure voltage isolation between the midi reception point and the microcontroller since Voltages greater than 5 volts can be damaging to the microcontroller.

### BRIEF INSIGHTS ON OPTOCOPULERS:

Optocouplers are electronic components that transfers electrical signals between two isolated circuits, they are very common for transmitting digital data. The left side of the optocoupler is connected to the sending circuit while the right side of the optocoupler is connected to the receiving circuit which allows isolation between two electrical communicating circuits.

Internally the left side of the optocoupler is comprised of a led, while the right side is comprised of a photosensor and a transistor that is activated when the photosensor senses some light from the led. The data sent is transcribed into the state of the transistor's collector that can be measured from one of the optocoupler's pins. Basically, by measuring the voltage state (HIGH or LOW) of the transistor's collector we retrieve the data that was sent.

### HOW IS A MIDI SIGNAL SENT AND RECEIVED:

A midi device sends the midi data through pin 4 and 5 on the midi cable **by tying pin 4 to a high voltage state always** and sending voltage pulses through pin 5 (*by voltage pulses we intend either high or low state , 1 or 0 etc..*), we should remember midi data is digital meaning they are just sequences of bits packed in group of bytes this will be explained further on.

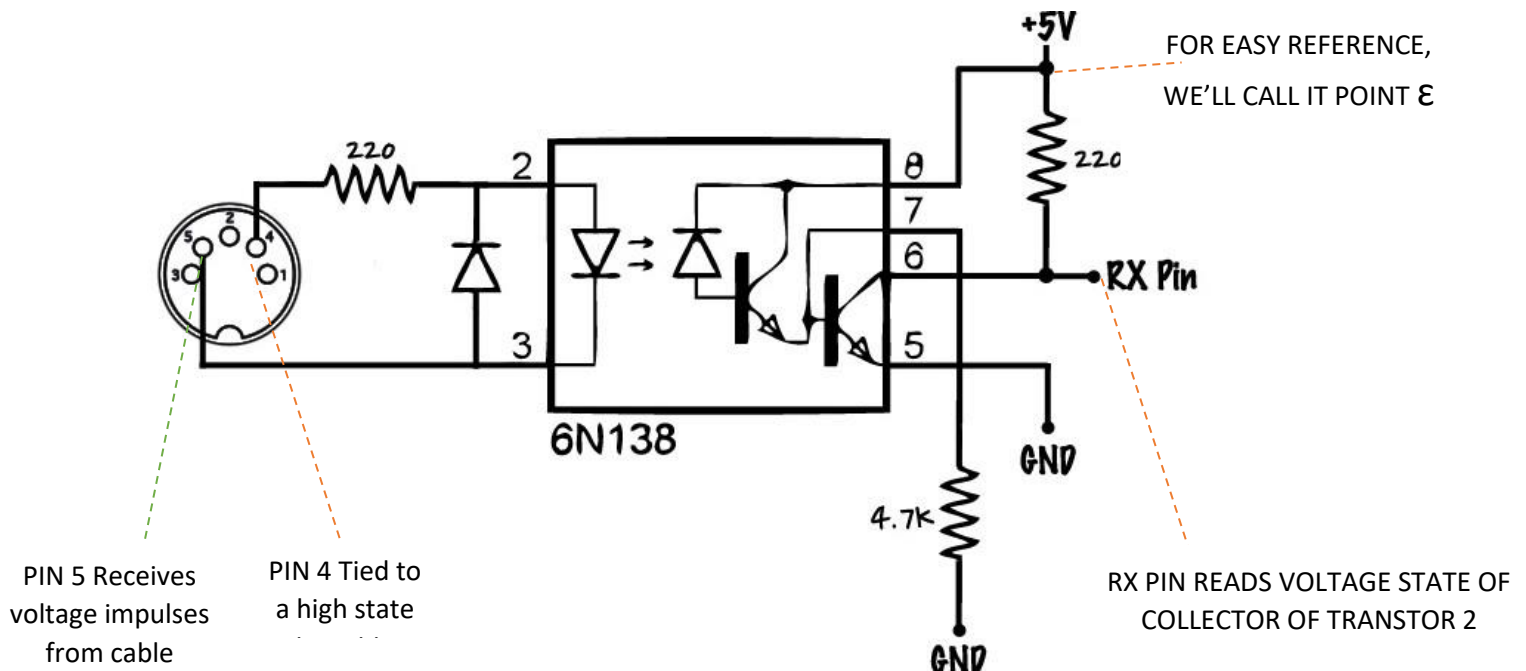
The cable from the keyboard having its pin 4 in a high state ties **pin 4 on the midi female** connector to a

## CDA 6316-001 PROJECT

high voltage and the pulses on pin 5 of the midi cable are transmitted to pin 5 of the female connector. Then the received bit is the **AND** operation between pin 4 and pin 5, meaning if pin 4 and pin 5 are both at high state, "1" is the bit received, if pin 5 is at a low state with pin 4 at a low state "0" is the bit received. And this is done at a rate of 31250 bits per second.

To correctly receive the midi data, an optocoupler is used to create voltage isolation since any high state above 5 voltage would damage the microcontroller.

Below is a clearer and more comprehensive schematic for figure 1.0, with the RX point indicating the point where the microcontroller collects the midi data.



Pin 4 of the female midi connector is always at a high state meaning it claims the role of VCC or voltage source. If the impulse on pin 5 is also at high state, then there is no current flow to the led inside the 6N138 Photo-Darlington Transistor Optocoupler since there is no voltage drop and the transistors inside the optocoupler are not activated **meaning there is no current flow from point ε** shown in the figure above, **therefore Point RX is at the same voltage of point ε.**

When pin 5 is at a low state, there is a voltage drop between pin 4 and PIN 5, therefore current flows to the led inside the 6N138 optocoupler which lights up to the photosensor that activates the transistors, the transistors in the optocoupler are activated, current flows from point ε to ground ( the current goes from point ε to **collector of transistor 2** then to **emitter of transistor 2** then to ground). Due to the current path which necessarily implies a voltage drop across the resistor, point RX is definitely at 0 volt potential meaning the microcontroller will read bit "0" . In summary when a low state is sent at pin 5 through the midi cable to the female midi connector, the microcontroller detects a digital low state.

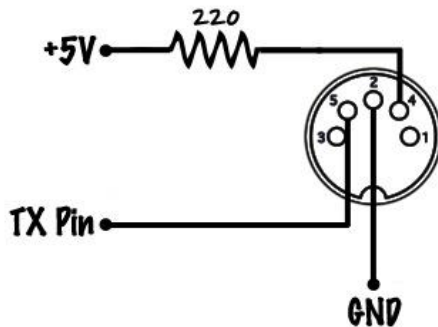
## CDA 6316-001 PROJECT

When pin 5 is tied to a high state by the midi cable, the microcontroller reads a digital high state.

### 2) MIDI TRANSMITTER ARCHITECTURE:

The objective of the midi transmitter is to transmit the **processed** midi signal or data received in input.

Below is the schematics for the midi transmitter :



Pin 4 on the female midi connector is constantly tied to a high state (5 volts) by the microcontroller, then once there is information to send meaning the bass note has been computed by the microcontroller based on the notes received in input, the bytes that represents the bass note is then transmitted from the TX pin on the microcontroller to the female midi connector on pin 5. If a midi device like a keyboard is connected to the female midi connector through a cable, the bass note data is sent by the microcontroller through the TX pin and it is received and played by the midi device.

### MIDI PROTOCOL:

To build the "PROCESSOR" the intermediary component which is the object between the transmitter and receiver, we need to first understand the midi protocol. The midi protocol operates at a baud rate of 31250 bits per second and The MIDI language is used to transmit real time information ( groups of bytes) . "Real time" means that each message is sent exactly at the moment and it must be interpreted with no delay by the receiver.

#### So how are Midi messages or Midi events organized ?

Traditional MIDI messages are sent as a time sequence of 3 bytes (1byte = 8 bits). The first byte is a **STATUS byte**, often followed by 2 **DATA bytes**. A **STATUS byte** has its 7<sup>th</sup> bit set to 1 and a **DATA byte** has its 7<sup>th</sup> bit set to 0. The status byte can be further be decomposed into two nibbles (1 nibble = 4 bits).

**STATUS BYTE** = 1XXXCCCC with 1XXX = note on OR off message, CCCC = channel number

**1<sup>st</sup> DATA BYTE** = 0YYYYYYY = Note number for example middle C<sub>3</sub> note number is 48

**2<sup>nd</sup> DATA BYTE** = 0ZZZZZZ = Velocity of the note, it describes how hard the note was pressed

The crucial point to know is that MIDI language does not define the sound itself, but only the sequence of instructions to create the sound in the receiver of the message.

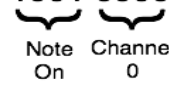
## CDA 6316-001 PROJECT

The status byte has its most significant bit (most significant bit is the first bit from left) set to 1 always because the first nibble can only assume two values : note on or off  
note on = 1001 which is 144 in decimal or 0F90 in hexadecimal  
note off = 1000 which is 128 in decimal or 0F80 in hexadecimal  
So a full status BYTE would of the form 1000CCCC or 1001CCCC.

Midi messages enable the transmission of only 128 types of notes pitches which approximate to  $128/12 = 10$  different octaves. This means the note values are in a range of (0-127) and can assume a maximum value of 127 and a minimum of 0. Since the maximum is 127 and 127 in binary is 01111111 meaning the most significant bit will always be set to 0, this also applies to 2<sup>nd</sup> data byte which represents the velocity information.

So a when a note is pressed the processor receives 3 bytes from the keyboard and when the note is released 3 further bytes are received.

### ***VISUAL EXAMPLE OF A MIDI MESSAGE :***

Status Byte		Data Byte 1	Data Byte 2
1001 0000		00111100	01111111
		Note Number 60 (C4)	Note Velocity 127

So when a note is pressed on the keyboard, the MIDI RECEIVER collects 3 bytes and passes them to the microcontroller which is the PROCESSOR.

The bytes are : 1) status byte : **note on + channel number** 2) **the number of the note pressed** 3) **The velocity of the note.**

When the note is released 3 further bytes are received and they are: 1) **status byte : note off + channel number** 2) **the number of note pressed** 3) **The velocity of the note.**

So after the 3 components that comprise the object we are trying to build have been well defined and assembled together, Verification was sought by making the med-basser send everything it receives in input to another target synthesizer. This can be achieved by using a smartphone as the target receiver ( a usb on the go cable would be needed) to receive any note played on the main keyboard. The code will be attached at the end of the report.

# CDA 6316-001 PROJECT

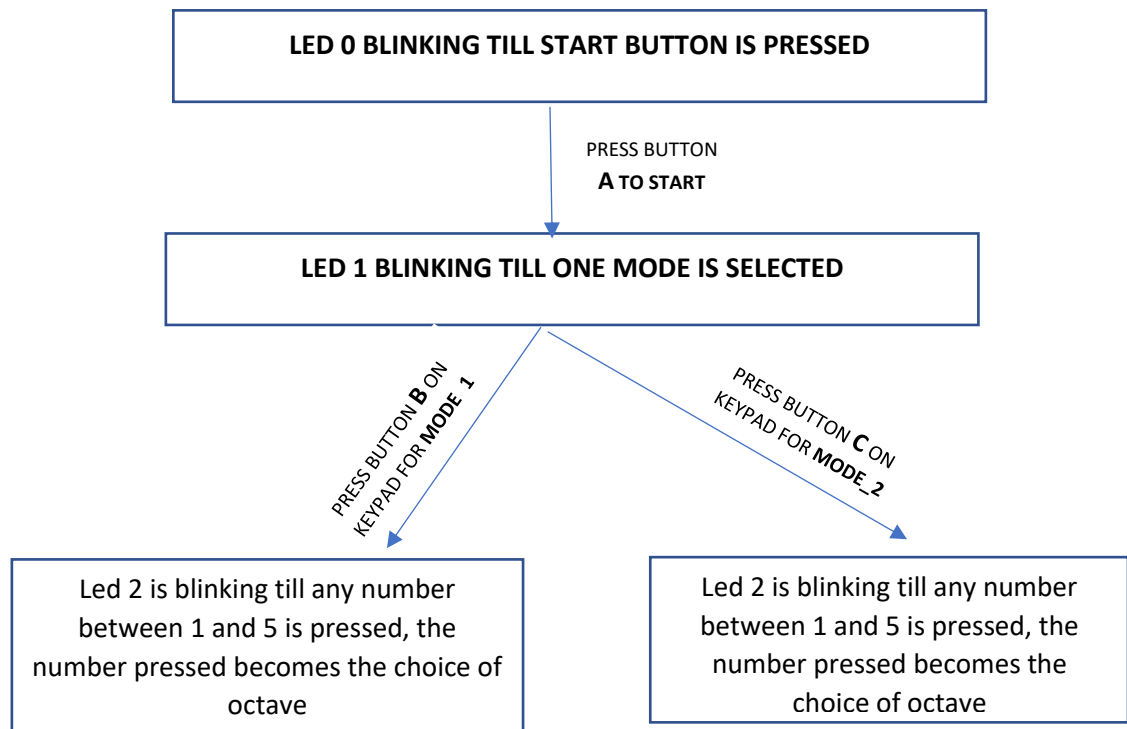
## THE PROCESSOR :

The processor is the logic behind the decision of the bass note, it decides what is sent by the midi transmitter. The processor has 3 basic fragments:

**The First segment** allows an interaction with the user, by allowing them to choose between modes of operation and the octave of the base note. This selection is done through the buttons on the keypad.

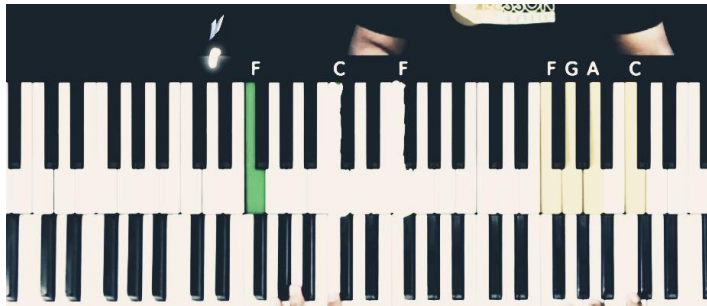
MODE 1 : JUST ONE BASS NOTE IS PLAYED DEPENDING ON OCTAVE CHOSEN

MODE 2 : MULTIPLE BASS NOTES WITH DOMINANT IS PLAYED DEPENDING ON OCTAVE CHOSEN



The option of having two modes comes from the observance of most piano players, where some chords are bass heavy because the bass notes are spread out in different octaves.

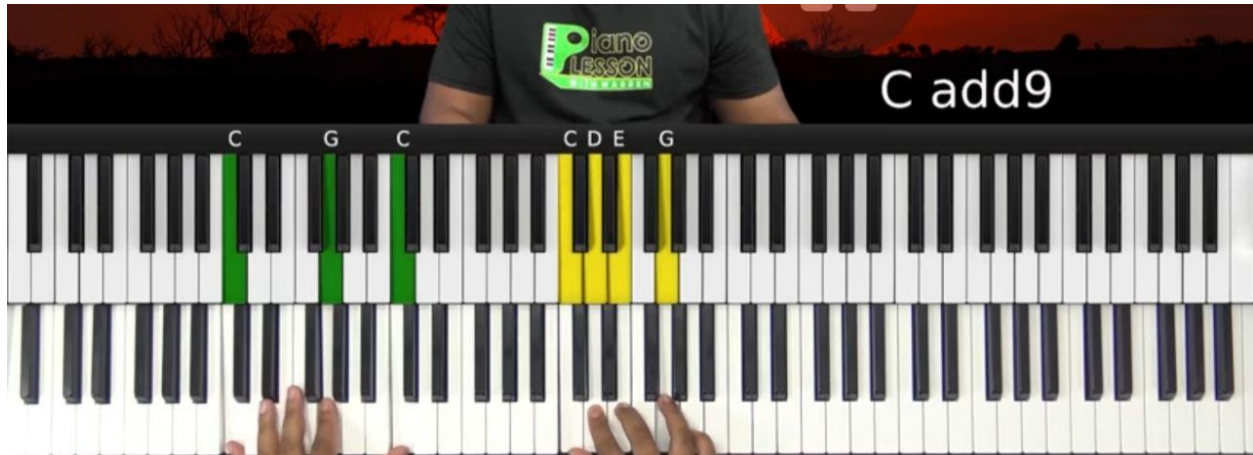
**MODE 1 : Just 1 bass note played to the F chord**





## CDA 6316-001 PROJECT

### MODE 2 : Multiple bass notes played in two different octaves to a C add9 chord



The second fragment of the processor represent a while loop that is only active when there is a **note on command**, every note played is saved in an array, when the notes played represent a chord, the bass note is computed with a chord identification algorithm I designed.

The chord identification algorithm is only able to identify **minor chords** and **major chords** and what it does is return the index of the array corresponding to the root note of the chord, for example if the chord played is a Gmajor chord (G D B), the algorithm returns the position in which the note G is in.

In this way we are able to recognize what chord was played through the root note, after this depending on the mode chosen the bass note is computed with the formula :

$$\text{bass\_note} = \text{root\_note} - (12 \times \text{octave\_selected})$$

If mode 1 was selected, bass\_note is a singular note, if mode 2 was selected, the multiple bass notes are computed and saved in array till *octave\_selected* decrements to zero.

The array or variable representing the bass note is then written to the microcontroller's serial port through the "Serial.write( ARRAY, array\_size)" function.

The third fragment represent a while loop that is only active when there is a **note off command**, it checks to see if the root note has been turned off. If the root note has been turned off, all base notes that are played are turned off.

# CDA 6316-001 PROJECT

## **SUMMARY & CONCLUSION:**

The objective of this project was to build a device that could help amputee individuals simulate the left or right arm when playing an electronic keyboard. This was done by incorporating an input apparatus to read the midi signals played by one arm. The data read in input is processed to decide the bass note in output. After the bass note (*or bass notes depending on the mode chosen*) have been established. It is sent in output to the *midi transmitter apparatus* that sends the signal to the electronic keyboard. All of these is realized in milliseconds to guarantee a real-time experience with no delay.

## REFERENCES

- 1) [http://www.personal.kent.edu/~sbirch/Music\\_Production/MP-II/MIDI/midi\\_protocol.htm](http://www.personal.kent.edu/~sbirch/Music_Production/MP-II/MIDI/midi_protocol.htm)
- 2) <https://www.cs.cmu.edu/~music/cmsip/readings/MIDI%20tutorial%20for%20programmers.html>
- 3) <https://ccrma.stanford.edu/~craig/articles/linuxmidi/misc/essenmidi.html>
- 4) <https://www.instructables.com/id/Send-and-Receive-MIDI-with-Arduino/>
- 5) <https://www.arduino.cc/reference/en/language/functions/communication/serial/>
- 6) <https://www.arduino.cc/reference/en/language/functions/communication/serial/readbytes>
- 7) <https://learn.sparkfun.com/tutorials/midi-tutorial/hardware--electronic-implementation>