GROUP NUMBER: 5

ROLL NUMBER 1: 20D170032 NAME 1: Reet Santosh Mhaske
ROLL NUMBER 2: 200260049 NAME 2: Shashwat Chakraborty

TITLE: Vaadyam (The Musical Instrument)

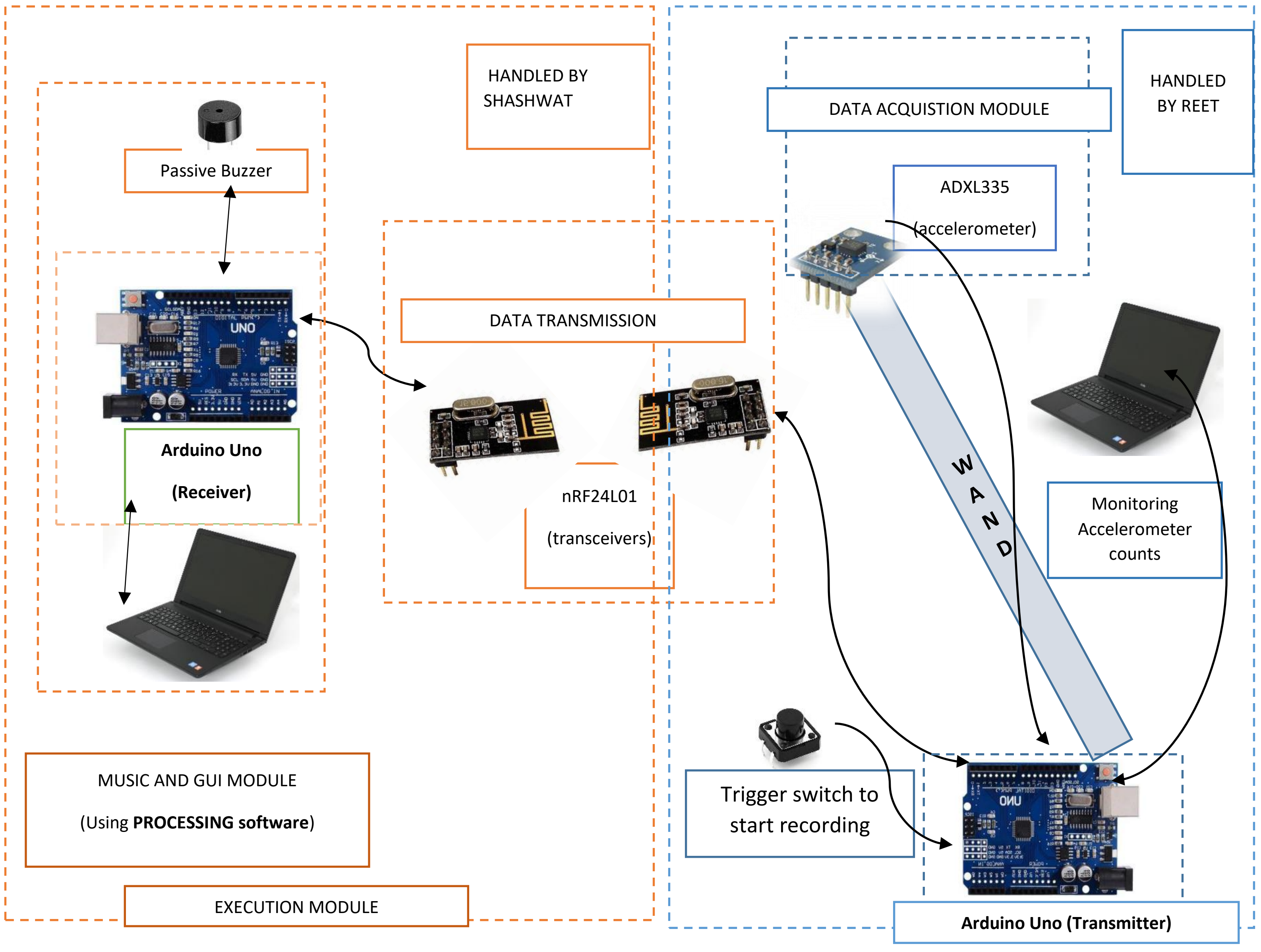
A musical wand that produces musical notes by its motion

ABSTRACT:

Wand has an accelerometer mounted on its tip. The number of times it flicks in 2 of the orthogonal directions is counted and sent via a transceiver to another distant Arduino. The receiver Arduino plays musical notes mapped to different number of strokes in different directions. The corresponding key is highlighted on the GUI application.

PROJECT DETAILS:

You must provide a block diagram of the major components of the project.
[1]
Use blocks to specify all sensors/input/output elements used. Give the details of the parts used, especially if they are commercially purchased parts (like ultrasonic sensors, LCD screens etc)



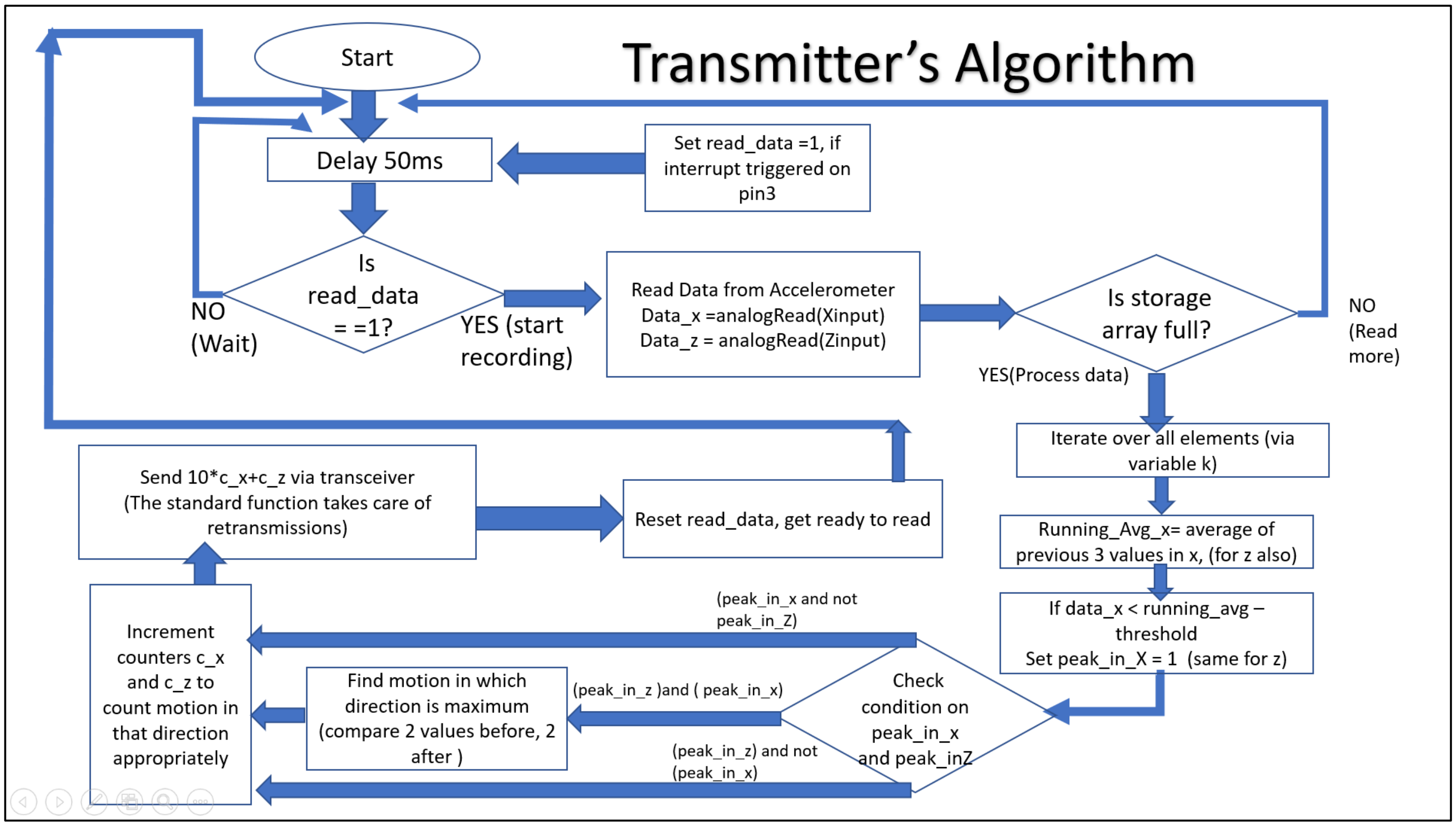
[2] No special breadboard circuitry was used, besides serving the purpose of connections. The trigger switch used was debounced using IC555, following the standard monostable circuit. Another extra Arduino was used for the purpose of providing a constant Vcc, without performing any special function.

[3]

The Arduino with the required algorithm should be in a separate block.

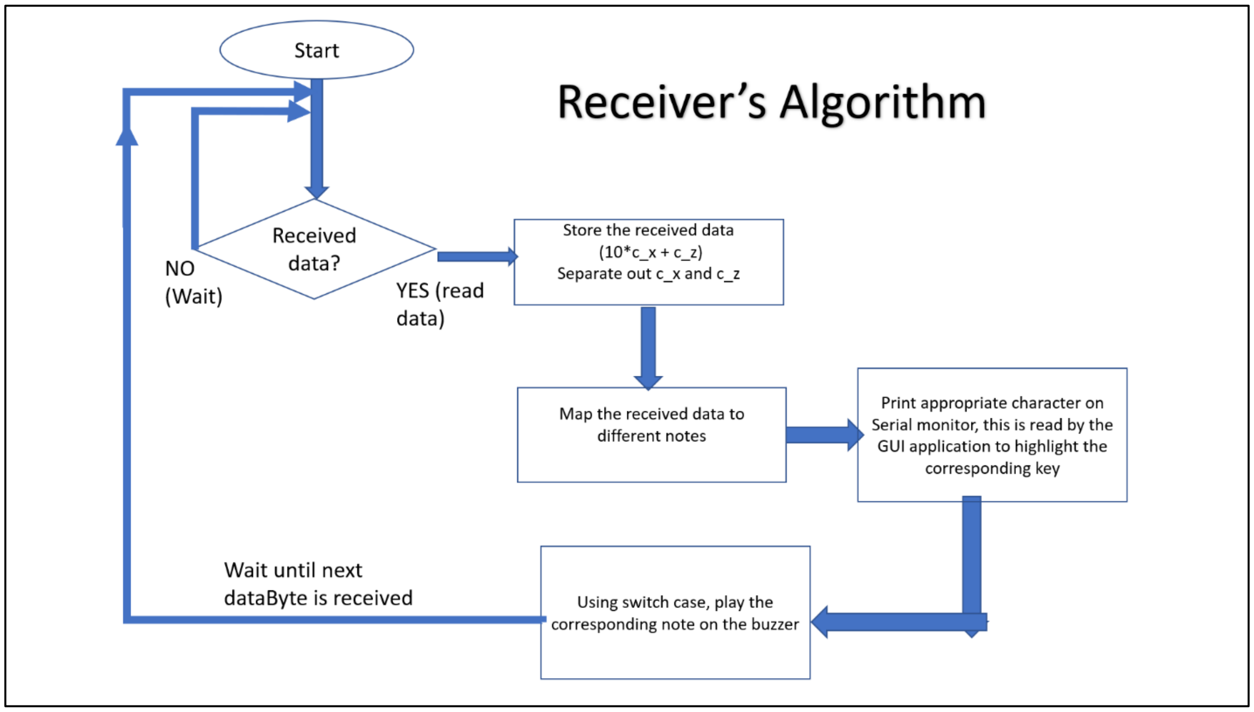
The algorithm flowchart must be shown here. Program code is to be included as an appendix to the report

Algorithm of Transmitter Arduino (on side of DATA ACQUISTION MODULE



Transmitter's Algorithm
Start
Delay 50ms
Set read_data =1, if interrupt triggered on pin3
Is read_data = =1?
NO (Wait)
YES (start recording)
Read Data from Accelerometer
Data_x =analogRead(Xinput)
Data_z = analogRead(Zinput)
Is storage array full?
NO (Read more)
YES(Process data)
Iterate over all elements (via variable k)
Running_Avg_x= average of previous 3 values in x, (for z also)
If data_x < running_avg – threshold
Set peak_in_X = 1 (same for z)
Check condition on peak_in_x and peak_inZ
(peak_in_x and not peak_in_Z)
(peak_in_z)and (peak_in_x)
(peak_in_z) and not (peak_in_x)
Find motion in which direction is maximum (compare 2 values before, 2 after)
Increment counters c_x and c_z to count motion in that direction appropriately
Send 10*c_x+c_z via transceiver (The standard function takes care of retransmissions)
Reset read_data, get ready to read

Algorithm of Receiving Arduino (on side of EXECUTION MODULE)



Receiver's Algorithm
Start
Received data?
NO (Wait)
YES (read data)
Store the received data (10*c_x + c_z) Separate out c_x and c_z
Map the received data to different notes
Print appropriate character on Serial monitor, this is read by the GUI application to highlight the corresponding key
Using switch case, play the corresponding note on the buzzer
Wait until next dataByte is received

Mark on your block diagram which group member was responsible majorly for working on which block. Writing 'both did everything' is not acceptable – surely you must have shared workload among the group members. The TA's have been tracking your progress.

Contributions:

Reet: Interfacing Accelerometer, detection of motion from accelerometer's data, assembly of wand,

Shashwat: Interfacing transceivers, development of GUI, interfacing passive buzzer , mapping musical notes

[4] If your project has significant analog circuits as part of the design, include LTSpice simulation circuit diagrams and simulation result plots of the analog component.

As mentioned, we have not used any complicated analog circuitry.

<u>MAIN COMPONENTS NEEDED TO BUILD THE PROJECT:</u>

Give an inventory of all the components you used to complete the project. If you needed to purchase some components other than the ones provided in your kit, please mention them separately.

Components:

Available Parts:

- i. 2 Arduino Uno's
- ii. Bread board ,Push button, jumpers, capacitors, registers

External purchase

- i. 1 passive Buzzer
- ii. 1 accelerometer ADXL335
- iii. 2 nrF24L01 transceivers

<u>RESULTS:</u>

Summarize the results of your project.

The key outcomes of the project were that we were able to achieve wireless data transmission between the transceivers and to process the accelerometer's data to make inferences about the wand's motion to a good degree of precession. We were able to arrive at the optimal values of the parameters used for finding the peaks in acceleration by repeated trials. We were also able to develop a GUI, and tie all things together. Thus, our project spanned 3 parts: accelerometer interfacing, transceiver interfacing and GUI development. The range of motion is restricted to vertical and horizontal (with the x direction of accelerometer pointing vertically). We demand that after each stroke, the user waits for about 0.5s before moving the wand again, for accurate counting.

Provide photos or links to video recording of your working project output. In the project demo and viva we expect a fully working end-result of your project work. But sometimes a last minute part failure may cause problems. In that case, we would like to see that your project worked at *some* time!Plus, a photogenic project output gets you a chance to get on the 'Projects Hall-of-Fame' poster board.

The following is the link to the drive with videos of our working project: https://drive.google.com/drive/folders/1-1yxKxoCjpCjf7uhc6MBlIbrraYCLUqy

We have also added the videos recorded of individual parts of the project. The final demo titled "Fianl Demo.MOV"

APPENDIX:

We have 3 parts of the code: The transmitter code, the receiver code and the GUI code that is run on the receiver's end on Processing

Code at Tranmitter end (the one connected to accelerometer):

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#define CE_PIN   9 // digital pin number of arduino to which CE of transceiver is connected
#define CSN_PIN 10 // digital pin number of arduino to which CSN of transceiver is connected
const byte slaveAddress[5] = {'R','x','A','A','A'};// the "pipe" address that is used to send the data
RF24 radio(CE_PIN, CSN_PIN); //  "radio" is now an "object" of class "RF24"
void setup() {
    Serial.begin(9600);
    Serial.println("Ready to transmit");  // a starting introductory message
 // THE FOLLOWING FUNCTIONS SET PARAMETERS OF THE TRANSCEIVER
    radio.begin(); // sets up the "radio" object
    radio.setDataRate( RF24_250KBPS );    // set the data rate (thru air) at 250KBps
    radio.setRetries(3,5); // delay, count
    radio.openWritingPipe(slaveAddress);  //  open a communication "pipe" over which data is sent
    analogReference(EXTERNAL);
//reference to the accelerometer ( we have connected AREF pin // to 3.3V( the arduino's 3.3 pin suffices))
    pinMode(3, INPUT_PULLUP);             // We are generating interrupt on pin3
    attachInterrupt(digitalPinToInterrupt(3),start_counting,RISING); }
   /* when pin3 goes HIGH, start_counting() function starts recording the data of the accelerometer
  for~5s( according to the delay values we have chosen, we can change this to other times, but all
  parameter then need to be tweaked )*/

//SEND FUNCTION
/* send() function sends the data by the transceiver, which we will use later for sending
accelerometer's data*/
void send(int dataToSend) {
    bool did_it_reach = radio.write( &dataToSend, sizeof(dataToSend) );
/* send the data and wait to receive an acknowledgement. This function takes care of retransmissions,
and returns False if no acknowledgement(ACK) as received from the receiver */
    // Now, if no ACK is received,even on multiple attempts, we print so to the monitor

    Serial.print("Data Sent "); // Print the data that was sent
    Serial.print(dataToSend);
    if (did_it_reach) { // if the ACK was received, the transmission was successful
        Serial.println(" ACK was received successfully");
    }
    else { // if ACK was not received,
        Serial.println(" No ACK received , transmission didn't go well");
    }}

//ACCELEROMETER'S VARIABLES
// Define the analog pins to which Accelerometer's input pins will be connected
const int xInput = A0;    // X pin to A0
const int yInput = A1;    // Y pin to A1 ( we don't use this in our code though)
const int zInput = A2;    // Z pin to A2
// We store the accelerometer's data in an array of size "data_size",
const int data_size =100;
volatile int  data_x[data_size]={0} ; //array to store accelereation in x direction
volatile int  data_z[data_size]={0} ; //array to store accelereation in z direction
int r=0; // index to fill data in the above arrays
volatile int read_data=0; /* read_data is a flag, which is set if the interrupt at pin3 is received
setting read_data flag tells the arduino to start recording the accelerometer's data untill the above
arrays are filled  Now, we use the above functions to actually read the data and transmit We are not
sending the accelerometer's data, but instead processing it and finding the no. of times the wand moved
in x or z direction and sending that*/
void loop()
 {delay(50);
  // waits for 50ms ( this is crucial is setting the sampling rate
  // all other parameters must be modified for good sampling outcome, if this rate is changed
  //We arrived at the optimal values by trial and error
  //DATA ACQUISITION
  if (read_data){// if the read_data flag is set, then read data
  data_x[r]= analogRead(xInput);// store data of x direction
  data_z[r]= analogRead(zInput);// store data of z direction
  r++;  // increment the index of the storage array
  if(r==100){ // if the array is filled with data, stop recording and now process the data
    count_data();  // function to process the data
    r=0; // set the array index to 0, so that next time we want to record, we start filling from index 0
    read_data=0; //reset the flag so that no more data is recorded
    }
      }}
// start_counting() is the fucntion attached to interrupt at pin3
// if pin3 has a rising pulse, we set read_data flag, as mentioned
void start_counting(){
  read_data=1; //set flag
  Serial.println("Started recording motion for 5s "); // 5s is the value it turns out to be for our
parameters
  //set index to 0
  r=0;}
/*we know that r ,read_data are being used in the main loop as well, (and must not be altered by an
interrupt)but we are sure in our case that interrupt implies suspending previous data and starting to
record from beginning */


//DATA PROCESSING
//count_data() function is the main data processing function
void count_data(){
int c_x=0;// stores no. of times the wand was flicked in x direction
int c_z=0;//stores no. of times the wand was flicked in z direction

// this algorithm is based on our observations of how the accelerometer readings change , to what extent
and on what motion
// accordingly, the parameters and threshhold below were decided by trial and error
for(int k =3;k<data_size-2;k++){ // parse the stored data
   int    running_avg_x=(data_x[k-2]+data_x[k-1]+data_x[k-3])/3;
// average of previous 3 values in x direction
   int    running_avg_z=(data_z[k-2]+data_z[k-1]+data_z[k-3])/3;
// average of previous 3 values in z direction
   // the current data must be a significant minimum of acceleration if it exceeds the previous average
by a threshold, which is 200 and 235
/* we don't detect a maximum, because we found that the peaks were well defined and separated as opposed
to the maximum, were multiple maxima can occur together , becaused of hand's unsteady motion */
   bool x_peaked =data_x[k]-running_avg_x<-200 ;  // experimentally set thresholds
   bool z_peaked =data_z[k]-running_avg_z<-235 ;
  if(x_peaked && !z_peaked) // if peak was detected in x, and not in z
  {c_x++; // increase count of x by 1
   k+=10;} // Ignore the next 10 sample points
    // On a sudden motion, there are transient spikes in the data, difficult to resolve
    //hence, on detecting a minimum, we ignore some of the next sample points
  // for the delay 50 we had set earlier in main loop, ignoring next 10 points seems to be a good choice
      if(z_peaked && !x_peaked) // similar logic , when a peak detected in z , not in x
  {c_z++;
   k+=10;}
/* It may happen that beacuse of unsteady motion, some component of acceleration appears in the other
direction*/
  /* We identify which direction the motion occured in, by finding the direction in which max change in
acc was seen*/
   if (x_peaked && z_peaked) // if peak is detected in both x and z
   { if(max(max(max(max(data_x[k-2],data_x[k-1]),data_x[k]),data_x[k+1]),data_x[k+2])>
max(max(max(max(data_z[k-2],data_z[k-1]),data_z[k]),data_z[k+1]),data_z[k+2]))
                  {c_x++;} // if the change in acc in x direction is more than z, based on the 5 sample
points centred around the current one,
                         // count this as motion in x

                  else{c_z++;}  // else this must be motion in z
        k+=10; }}               // skip the next 10 sample points

    int count_x_z =10*c_x + c_z;
/* instead of returning c_x,c_z, encode them into one variable and return we are assured by the
constraints on the parameters that the storage array fills before the wand is moved more than 6 times*/
   // hence, c_z will never exceed 10
   // Using the transceiver's send function, send the encoded answer to the receiver
    send(count_x_z);
    r=0;} // set storage array's index to 0
```

Code for receiver

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#define CE_PIN   9 // CE of transceiver connected to digital pin 9
#define CSN_PIN 10 // CSN of transceiver connected to digital pin 10
// DEFINE VARIOUS FREQUENCY NOTES
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define END -1
// DEFINE THE FREQUENCY NOTES OF THE OCTAVE BEING USED
// WE CAN USE ANY OTHER OCTAVE FROM THE ABOVE FREQUENCIES AS WELL
int sa_m = NOTE_CS4;
int re = NOTE_DS4;
int ga = NOTE_F4;
int ma = NOTE_FS4;
int pa = NOTE_GS4;
int dha = NOTE_AS4;
int ni = NOTE_C5;
int sa_t = NOTE_CS5; //HIGHER SA
const byte thisSlaveAddress[5] = {'R','x','A','A','A'};
/* THE address of the reading "pipe" from which data will be received
(is same as that of the pipe to which the sender is transmitting)*/
RF24 radio(CE_PIN, CSN_PIN); // "radio" is an object of class "RF24"
int dataReceived; // the variable to store the receieved data
bool  newDataReceived = false; // flag to indicate that new Data was received
void setup() {
    Serial.begin(9600);
    radio.begin(); // setup the radio object
    radio.setDataRate( RF24_250KBPS ); // bitrate thru air is 250KBps
    radio.openReadingPipe(1, thisSlaveAddress);
    //  There are 5 reading pipes, of which we use the 1st one, and label it with this address
    radio.startListening();
    // start listening the pipe
}
int interval = 500; // Play the note for 500 ms,
void loop() {
    getData();
    // if some data is received, getData writes it to the variable "dataReceived"
    //showData();
    /* showData() is used for debugging,it prints the data that was received */
    int receivedByte = dataReceived;
    // Assuming 10*c_x + c_z was received , we map c_x and c_z to different notes
    // receivedByte/10 = c_x and receivedByte%10 = c_z
 /*map the received data, then play the corresponding note, and send a char to the monitor,
  this character is read by the GUI application running on Processing, which highlights the
corresponding key*/
    switch (receivedByte)
    { case 0: Serial.write('I');  // no motion
                                  //write the char "I" to monitor
              break;
      case 1:  Serial.write('s');  //1 horizontal motion
                                  //write the char "s" to monitor
              tone(3, sa_m, interval); // play the note
              break;
     case 2:  Serial.write('r');    // 2 horizontal strikes
              tone(3, re, interval); // play note
              break;
     case 3:  Serial.write('g'); // 3 horizontal strikes
              tone(3, ga, interval); // play note
              break;
     case 10:  Serial.write('m'); // 1 vertical strikes
              tone(3, ma, interval); //play note
              break;
     case 20:  Serial.write('p'); // 2 vertical strikes
              tone(3, pa, interval); // play note
              break;
     case 30:  Serial.write('d'); // 3 vertical strokes
              tone(3, dha, interval); // play note
              break;
     case 11:  Serial.write('n'); // 1 Vertical and 1 horizontal motion
              tone(3, ni, interval); // play note
              break;
     case 21:  Serial.write('S'); // 2 vertical and 1 horizontal motion
              tone(3, sa_t, interval); // play note
              break;
    }}
/* the getdata()function writes the received data into the global variable dataReceived*/
void getData() {
    if ( radio.available() ) { // if bytes are available for reading
        radio.read( &dataReceived, sizeof(dataReceived) );
        // read data from the FIFO buffer of th transceiver
        newDataReceived = true; // set flag to show new data was received
    }}
/*showData() is used for debugging
It prints the dataByte received*/
void showData() {
    if ( newDataReceived == true) {
      // if the newDataReceived flag was set, then some new data was received, show this data
        Serial.print("Data received ");
        Serial.println(dataReceived);
        Serial.println("No. of counts in x:");
        Serial.println(dataReceived/10);
        Serial.println("No. of counts in z:");
        Serial.println(dataReceived%10);
        newDataReceived= false; // indicates that this data was read, so reset th flag
    }}
```

Code for GUI (to be run on Processing)

```
import controlP5.*; //import ControlP5 library
import processing.serial.*;
import ddf.minim.*;
Serial port;
Minim minim;
AudioPlayer player1, player2;
ControlP5 cp5; //create ControlP5 object
// defined the keys
int sa_m_detect = 0, re_detect = 0, ga_detect = 0, ma_detect = 0, pa_detect = 0, dha_detect = 0,
ni_detect = 0, sa_t_detect = 0;
PShape sa_m, re, ga, ma, pa, dha, ni, sa_t;
PShape border_rect;
void init(){
    sa_m.setFill(color(0, 0, 255));
    shape(sa_m, 100, 100);
    re.setFill(color(0, 0, 255));
    shape(re, 100, 100);
    ga.setFill(color(0, 0, 255));
    shape(ga, 100, 100);
    ma.setFill(color(0, 0, 255));
    shape(ma, 100, 100);
    pa.setFill(color(0, 0, 255));
    shape(pa, 100, 100);
    dha.setFill(color(0, 0, 255));
    shape(dha, 100, 100);
    ni.setFill(color(0, 0, 255));
    shape(ni, 100, 100);
    sa_t.setFill(color(0, 0, 255));
    shape(sa_t, 100, 100);



    sa_m_detect = 0;
    re_detect = 0;
    ga_detect = 0;
    ma_detect = 0;
    pa_detect = 0;
    dha_detect = 0;
    ni_detect = 0;
    sa_t_detect = 0;
}
void setup(){ //Same as setup in arduino
  size(900, 800);                          //Window size, (width, height)
  port = new Serial(this, "COM5", 9600);
  // Setup the physical layout of the keys on GUI
border_rect = createShape(RECT, 50, 350, 800, 300);
  sa_m = createShape(RECT, -25, 300, 50, 200);
  sa_m.setFill(color(0, 0, 255));
  re = createShape(RECT, 75,300,50,200);
  re.setFill(color(0, 0, 255));
    ga = createShape(RECT, 175,300,50,200);
  ga.setFill(color(0, 0, 255));
    ma = createShape(RECT, 275,300,50,200);
  ma.setFill(color(0, 0, 255));
    pa = createShape(RECT, 375,300,50,200);
  pa.setFill(color(0, 0, 255));
    dha = createShape(RECT, 475,300,50,200);
  dha.setFill(color(0, 0, 255));
    ni = createShape(RECT, 575,300,50,200);
  ni.setFill(color(0, 0, 255));
    sa_t = createShape(RECT, 675,300,50,200);
  sa_t.setFill(color(0, 0, 255));}
int inByte;
void draw(){  //Same as loop in arduino
   background(255); //Background colour of window (r, g, b) or (0 to 255)
  textSize(50);
  fill(0,0,255);
// Display background text
  text("VAADYAM", 250, 100);
  textSize(30);
  fill(0,255,255);
  text("Move the wand and produce melodious music", 100, 150);
  textSize(30);
  fill(0,0,0);
  text("Piano Keys", 350, 320);
  shape(border_rect);
  shape(sa_m, 100, 100);
  shape(re, 100, 100);
  shape(ga, 100, 100);
  shape(ma, 100, 100);
  shape(pa, 100, 100);
  shape(dha, 100, 100);
  shape(dha, 100, 100);
  shape(ni, 100, 100);
  shape(sa_t, 100, 100);
  while (port.available() > 0) {
    inByte = port.read();
    //sa
    if(inByte == 's')
    {print('s');
    if(sa_m_detect==0){
      //re.setFill(color(0, 0, 255));
      //shape(re, 100, 100);
      init();
      sa_m.setFill(color(255, 0, 0));
      shape(sa_m, 100, 100);
      sa_m_detect=1;}
    else if(sa_m_detect==1)
    { init();}}
   else if(inByte == 'r')
    {print('r');
    if(re_detect==0){init();
      re.setFill(color(255, 0, 0));
      shape(re, 100, 100);
      re_detect=1;}
    else if(re_detect==1)
    {init();}}

 // ga
```

```
  else if(inByte == 'g')
    {print('g');
    if(ga_detect==0){init();
      ga.setFill(color(255, 0, 0));
      shape(ga, 100, 100);
      ga_detect=1;}
    else if(ga_detect==1)
    {init();}}
  else if(inByte == 'm')
    {print('m');
    if(ma_detect==0){
      init();
      ma.setFill(color(255, 0, 0));
      shape(ma, 100, 100);
      ma_detect=1;}
    else if(ma_detect==1)
    {init();}}
   //pa
  else if(inByte == 'p')
    {print('p');
    if(pa_detect==0){
      init();
      pa.setFill(color(255, 0, 0));
      shape(pa, 100, 100);
      pa_detect=1;}
    else if(pa_detect==1)
    {init();}}
   //dha
   else if(inByte == 'd')
    {print('d');
    if(dha_detect==0){
      init();
      dha.setFill(color(255, 0, 0));
      shape(re, 100, 100);
      dha_detect=1;}
    else if(dha_detect==1)
    {init();}}
   //ni
   else if(inByte == 'n')
    {print('n');
    if(ni_detect==0){
      init();
      ni.setFill(color(255, 0, 0));
      shape(ni, 100, 100);
      ni_detect=1;}
    else if(ni_detect==1)
    { init();}}
   //sa_t
   else if(inByte == 'S')
    {print('S');
    if(sa_t_detect==0){
      init();
      sa_t.setFill(color(255, 0, 0));
      shape(sa_t, 100, 100);
      sa_t_detect=1;}
    else if(sa_t_detect==1)
    { init();}}
   //otherwise
   else if(inByte == 'r')
    {print('X');
    init();
    } }}
```

References:

We have written with the codes for accelerometer and the GUI with very little help from internet references. However, we would like to mention that we borrowed some parts of the transceiver code from the following site: https://forum.arduino.cc/t/simple-nrf24l01-2-4ghz-transceiver-demo/405123 to establish communication between the transceivers