Final Report ECE 5

Dayita Ray, Srinivasan Arumugham, Rachit Gupta

Device Functionality:
The player presses the button corresponding to the correct tile as seen on the screen and it records how fast a player completes a sequence of tiles. If the player clicks on the wrong button, the game will end. The longer the player is able to keep clicking on the correct buttons, the higher their score will be! The score is given by the amount of time it takes the player to finish the game, with under 5 seconds being the quickest the player can finish in and under 40 seconds being the slowest.

Hardware Components Used:

- Arduino Board
- VGA Display
- VGA Breakout Board
- Push Buttons
- Piezo Buzzer
- Jumper Cables
- Capacitors
- Resistors
- Perforated Boards
- Cardboard Box

Design Timeline:

1. Session 1: Purchased components needed to build the project
2. Session 2: We assembled the circuit on a breadboard and connected it to a VGA monitor. We used prewritten code to test out if the connections worked and if we could actually run the display on an Arduino.
3. Session 3: We started to program, trying to draw basic rectangles on different parts of the screen and tried to make them move in order to gain a better grasp on the VGAx library.
4. Session 4: We also tested the button input via using a capacitor network. And made sure we were able to record the button presses fast enough.
5. Session 5: We debugged the code and made an outer casing to hold all the components. We soldered the VGA connections onto a perfboard. We also used a box to create a container for the buttons to be pressed to play the game.

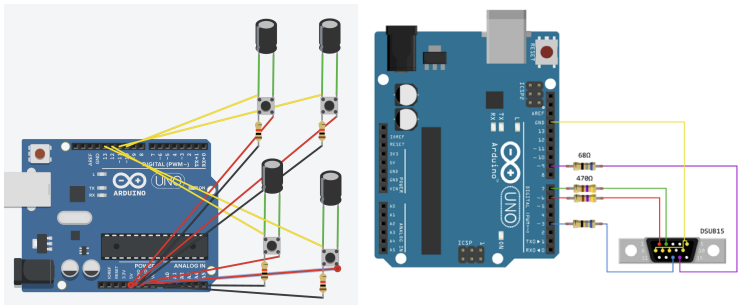
Software Design:

```
#include <VGAX.h>
#include <math.h>

//font generated from 2BITFONT - by Sandro Maffiodo
#define FNT_UFONT_HEIGHT 6
#define FNT_UFONT_SYMBOLS_COUNT 95

//data size=570 bytes
//the next 94 lines contain sprites for all the ascii characters that are available
for the library to use
const unsigned char fnt_ufont_data[FNT_UFONT_SYMBOLS_COUNT][1+FNT_UFONT_HEIGHT]
PROGMEM={...};

//text for the different screens
static const char str0[] PROGMEM="Welcome to \n      PIANO TILES!\n\n Press any button
to play";
static const char str1[] PROGMEM="Thanks for playing\n\npress any button to \nplay
again>";
static const char strS[] PROGMEM="GREAT JOB!\nYOU TOOK LESS THAN \n5 SECONDS!";
static const char strA[] PROGMEM="Good Job!\nYou took less than \n10 seconds!";
static const char strB[] PROGMEM="Good Try!\nYou took less than \n20 seconds.";
static const char strC[] PROGMEM="You should Try again!\nYou took more than \n20
seconds.";
static const char strD[] PROGMEM="You took more than \n30 seconds.";
static const char strE[] PROGMEM="You did terrible!\nYou took more than \n40 seconds";

//initialize a VGAx object vga and a bute for the background color
//black is 00
VGAX vga;
byte bgColor = 00;

//initialize a boolean array to store the states of the buttons at any instance in
time
bool pins[4] = {false,false,false,false};

//a "song" consisting of 50 values for each column at each instatnce in time
//the song has a set of 6 4s at the end to prevent indexing errors
const int song[50 + 6] = {3, 1, 2, 0, 1, 2, 3, 0, 2, 0, 1, 3, 2, 0, 1, 3, 1, 3, 0, 2,
3, 2, 3, 0, 1,2, 0, 1, 0, 3, 2, 0, 3, 1, 2, 0, 1, 2, 0, 3, 2, 0, 1, 3, 1, 0, 2, 0, 3,
2, 4, 4, 4, 4, 4, 4};

//the rect function acts as a wrapper for VGAX.fillrect and takes a collumn, a y
position and a color as input
void rect(int x, int y, byte color) {
 vga.fillrect(x * 30, y, 30, 10, color);
}

// function draws the starting screen on the display.
void drawMenu(){
 vga.printPROGMEM((byte*)fnt_ufont_data, FNT_UFONT_SYMBOLS_COUNT, FNT_UFONT_HEIGHT, 3,
1, str0, 10, 10, 01);
}

// function draws the ending screen on the display.
void drawEnd(){
 vga.printPROGMEM((byte*)fnt_ufont_data, FNT_UFONT_SYMBOLS_COUNT, FNT_UFONT_HEIGHT, 3,
1, str1, 10, 10, 01);
}
```

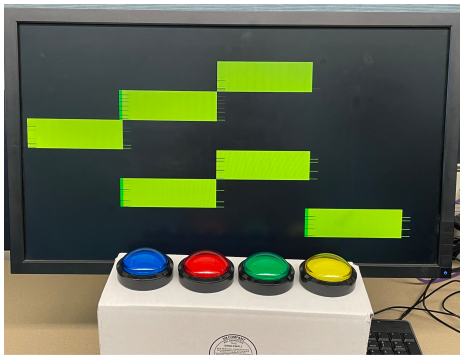
```
// in the setup loop,
 // intialising the VGA library.
 // setting pins 10-13 as input to record button pressed.
void setup() {
 vga.begin();
 vga.clear(bgColor);
 pinMode(10, INPUT);
 pinMode(11, INPUT);
 pinMode(12, INPUT);
 pinMode(13, INPUT);

}

//processInputs checks each of the button pins and stores their state into the
corresponding index in the pins array
void processInputs() {
 for(int i = 0; i < 4; i ++){
   pins[i] = digitalRead(10 + i);
 }
}


int state = 1;
int startTime;
int time;
int current_y = 0;


void loop() {
 processInputs();

 //state 1 prints the starting screen
 if(state == 1) {
   drawMenu();
 }

 // if any button is pressed in state 1, it moves it into state 2 which is a
transition state
 if( state == 1 && (pins[0] || pins[1] || pins[2] || pins[3])){
   state++;
 }
 // in state 2, when the button is unpressed, the state is incremented.
     //StartTime is Recorded, the variable current_y is initialised,
 if(state == 2 && !(pins[0] || pins[1] || pins[2] || pins[3])){
   startTime = vga.millis();
   vga.clear(00);
   state++;
   current_y = 0;
 }

// state 3 is the MAIN GAME
 if(state == 3) {
   // at any given time, there are 6 rectangles on the screen.
   // we use a for loop to print any given state.
   for(int i = 0; i < 6; i++){
     rect(song[current_y + i], 50 - 10 * i, 11);
   }

   // this checks if the button corresponding to the rectangle at the bottom of the
screen is pressed.
   // if yes, it increments y and clears the screen.
   if(pins[song[current_y]]){
     current_y++;
```

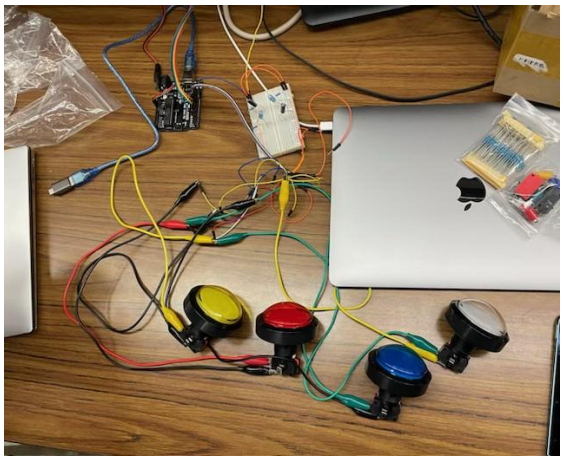
```
      vga.clear(00);
    }

    // when the the song ends, state is incremented to state 4
    if(song[current_y] ==4){
      state++;
    }
  }

// in state 4, Final Time is calculated and the state is incremented to 5
  if(state == 4){
    time = vga.millis() - startTime;
    state++;
  }

  //in state 5, after the player releases all buttons,
    // it displays how good or bad the player did!
    // the state is incremented.
  if(state == 5 && !(pins[0] || pins[1] || pins[2] || pins[3])){
    vga.clear(00);
    // the switch statement determines which message to display depending on the time
it took the user to complete the game
    switch(time / 5000) {...}
   state++;
  }
   //state 6 is a transition state
  if(state == 6 && (pins[0] || pins[1] || pins[2] || pins[3])){
    state ++;
  }
  // in state 7,  it prints the ending screen.
  if(state == 7 && !(pins[0] || pins[1] || pins[2] || pins[3])){
    vga.clear(00);
    drawEnd();
    state++;
  }
  //state 8 is a transition state
  if(state == 8 && (pins[0] || pins[1] || pins[2] || pins[3])){
    state ++;
  }
  //redirects theplayer to state 1 (play again) when the button is unpressed
  if (state == 9 && !(pins[0] || pins[1] || pins[2] || pins[3])){
    vga.clear(00);
    state = 1;
  }
}
```

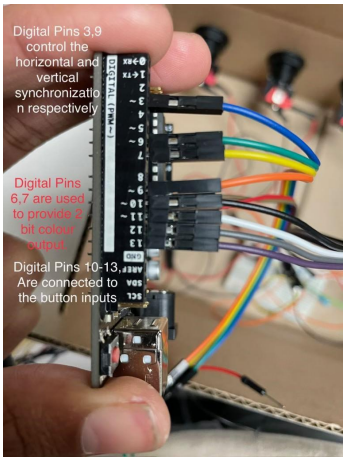
Circuit Schematics:



Circuit Prototype:







Testing:

We tested out device every step of the project, we did extensive testing on the button input using the serial plotter to ensure our signal was clean. We also did many tests with the VGA monitor, displaying various patterns and using trial and error to find the best way to display moving graphics on the screen without glaring visual artefacts.