# CS 362: Milestone 4

Due on March 29, 2024 at 11:59pm

*Professor Troy 11:00am*

**John Ezra See**
**Ryan Magdaleno**
jsee4@uic.edu
rmagd2@uic.edu

## Group Information

Group 62: VSRG-UNO-R3
Names:

1. Ryan Magdaleno: rmagd2@uic.edu
2. John Ezra See: jsee4@uic.edu

---

## Abstract

This project introduces an original Arduino/C++ endeavor aimed at enhancing the gaming experience through innovative hardware and software integration. Utilizing Arduino microcontrollers and a photoresistor array, it offers physical interaction with rhythm games, synchronizing gameplay with real-time data processing. The system manages communication between components and supports customization preferences. Through this project, users can enjoy an immersive and customizable VSRG experience, demonstrating creativity and innovation in Arduino/C++ technology.

---

## 1 Project Idea

Our project idea centers around enhancing the experience of playing Vertical Scrolling Rhythm Games (VSRGs), particularly focusing on key presses. We plan to utilize two Arduino microcontrollers to manage two displays and a photoresistor array. One Arduino will control the displays, showcasing both the keys pressed per second and the key press strokes. This data will be calculated and sent via serial communication to a computer-side program. We intend to update the display with the keys per second value every 20 milliseconds on a 16x2 LCD screen. Additionally, we will transmit a binary value representing the keys pressed, also updating every 20 milliseconds.

For gameplay, we've chosen Osu Mania due to its easily parseable .osu file format, which contains hit timings for beatmaps. Although Osu Mania supports maps of various key counts, we will focus on maps ranging from 4 to 7 keys. We've changed our original idea of using a solenoid array to a photoresistor array because there were some complexities in the solenoid circuit. Now one Arduino will read the incoming notes and check if it's in front of the photoresistor. The photoresistor array Arduino will send the binary encoding of the current keys pressed to a computer side program. The computer side program will then press those keys. The computer program will also continuously monitor these key presses, calculating both keystrokes and keys per second, and sending the accurate values to the display Arduino. For the Arduino managing keystrokes, we will employ a Nokia 5110 LCD to display the pressed keys, animating them upwards for visual feedback.

---

## 2 Initial Project Design

One Arduino will read the screen with photoresistors and send the current notes to press, then it will be sent to a computer side program. The other Arduino will receive keyboard press information, including keys per second and recent key presses. The computer-side program will start both Arduinos upon pressing key H and stop them upon pressing key J. A Serial connection will be transmitted from the photoresistor Arduino to the display Arduino (via RX/TX pins), terminating both Arduinos simultaneously.

---

## 3 Arduino Communication

For communication, we'll utilize a computer-side program to send the necessary data to each Arduino as required. The photoresistor Arduino will send hit data, while the display Arduino will receive keys per second (KPS) and keystroke data.

Initiating and stopping both Arduinos will be controlled by the computer-side program. To begin operation, the key H must be pressed. When this key is pressed, the computer-side program will send the photoresistor Arduino a signal (0xFFFF), which will then be forwarded to the display Arduino via the TX and RX pins. Termination will follow a similar procedure.

---

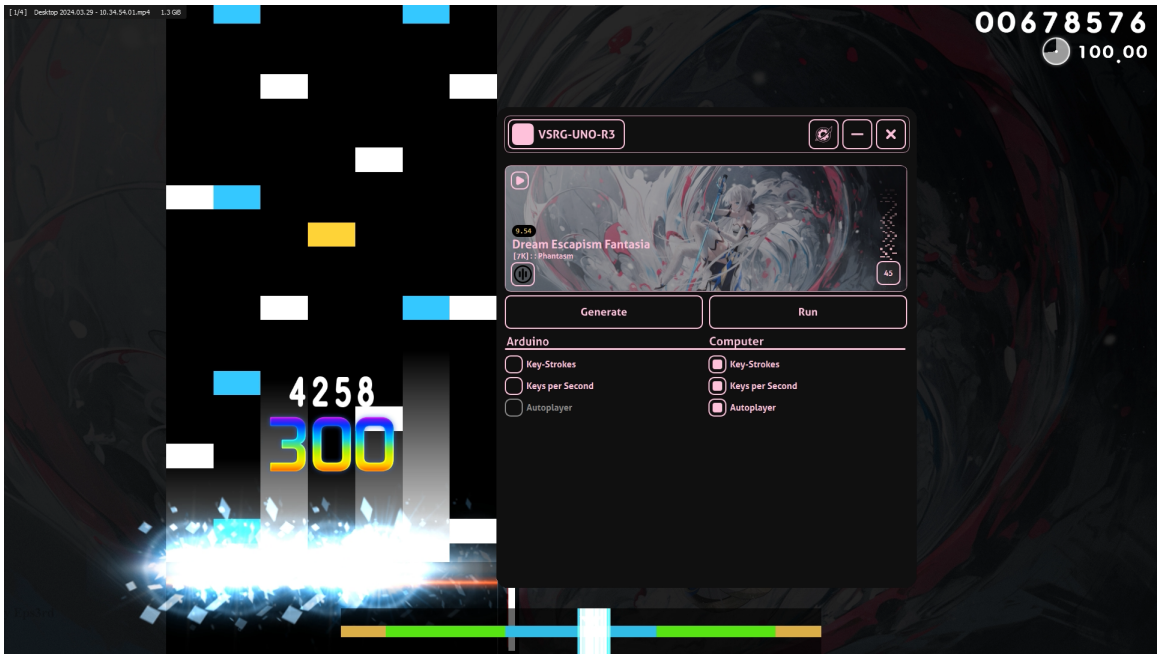
## 4 Expected Inputs/Outputs

**Expected Inputs**

1. **Keyboard Input:** The user would input commands through the keyboard. For example, pressing the key H to start the system and the key J to terminate it.
2. **Beatmap Hit Data (.osu file):** This data would be inputted into the computer-side program, to provide the timing and sequence of hits in the osu beatmap.
3. **Note Light (photoresistors):** The photoresistors will be reading the falling notes, it requires the note to be in front of the photoresistor.

**Expected Outputs**

1. **Display Data:** The display Arduino would output data to the connected display, showing information such as keys per second and recent key presses. This provides visual feedback to the user. KPS on a 16x2 LCD, key strokes on a Nokia 5110 LCD.
2. **Status Indicators:** The computer-side program may output status indicators or messages to inform the user about the system's operation, such as when it has started or terminated successfully.
3. **Overall GUI program:** The computer-side program has a GUI, it will display many options to the user about what is going on, what is being transmitted, etc. It's a work in progress currently.




---

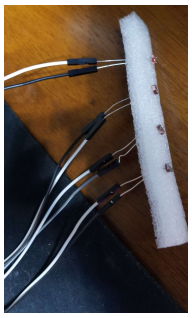
## 5 Original Work Justification

This project presents an original Arduino/C++ endeavor through its fusion of hardware and software elements to enhance the Osu Mania gaming experience. By integrating custom hardware components like Arduino microcontrollers and a photoresistor arrays, the system goes beyond typical software-based modifications, allowing for physical interaction with the game. It processes real-time data, including beatmap hit timings and key presses, to synchronize physical actions with gameplay, showcasing advanced programming techniques in C++ (multi-threading, chrono timing, windows serial handles). Managing communication and synchronization between multiple components, including the computer-side program, Arduinos, photoresistor array, and displays, demonstrates a sophisticated understanding of system architecture and programming principles. Overall, this project stands out as an original and innovative Arduino/C++ project, offering a creative VSRG gaming experience. A GUI is used to facilitate this.

---

## 6 How to Build the Project

1. Head over to https://github.com/typeRYOON/VSRG-UNO-R3
2. You can build from source following the instructions I set in the description or just get an already built binary from the Release section. If it's there (still working on making a version 1).
3. For Arduino hardware please look at the diagrams below for what wiring you need. Because the computer side program allows using 0, 1, or both Arduino configurations, build whichever ones you'd like.
4. For the photoresistor Arduino, put your photoresistors through something like Foam and tape it to the osu mania game screen, it needs to read incoming notes. Please download the osu skin from the res GitHub folder, it's optimal for this program.



---

## 7 How to Use the Project

1. Connect both Arduinos to the computer you're going to use. Note the port name.
2. On the project's Github repository, get the Arduino ino files from the Arduino folder.
3. Upload the ino program to each Arduino.
4. You must have Osu installed and running before you open the GUI, it needs to latch to the Osu process to read memory signatures (for GUI purposes).
5. Make sure to install the Noto Sans JP font from the Github on your system.
6. Before launching the GUI, enter Osu and set your osu mania 4K, 5K, 6K, 7K keys, the program needs a valid set of keys to work with.
7. Launch the GUI and go to the settings page (gear icon at top), set the Arduino settings with the port selector.
8. To run a configuration, select the beatmap in game and click on the GUI's generate button along with the features you want to run.
9. Assuming you're only using Arduino features, press H to start the two Arduinos.
10. If you're using the computer autoplayer feature, press H on the first note.
11. To terminate early, press J. You can also wait until the map finishes and the Arduinos will terminate for you.
12. An extra note, if you're using the Arduino autoplayer feature, please use the osu skin in the res GitHub folder.

An extra extra note, the GUI binary may not be on GitHub, everything is still a work in progress. Some things like the skin may also be missing. But this is the general program flow that we intend to have. We expect to implement everything around 4/15.

---

## 8 Timeline of Development

Completed Items:

1. **Week 9 (03/11/24):**
   Ryan finished up the display Arduino code, and made a CLI for interacting communicating with the Arduinos.
   John's circuit catches on fire after connecting a 950 mA solenoid with a 600 mA transistor.
2. **Week 10 (03/18/24):**
   Ryan and John finished the photoresistor circuit/general code.
3. **Week 11 (03/25/24):**
   Ryan learns Qt6 over spring break and makes a working GUI.

Work left to do:

1. **Week 12 (04/01/24):**
   Ryan will try finishing the GUI, otherwise will help John.
   John will try implementing a 8to1 mux for 7K (7 photoresistors).
2. **Week 13 (04/08/24):**
   Ryan will try finishing the GUI, otherwise will help John.
   John will try implementing a 8to1 mux for 7K (7 photoresistors).
   Make final stand for both Arduinos to sit on.
3. **Week 14 (04/15/24):**
   Create presentation slides and present on 04/15/24 in lab.
4. **Week 15 (04/22/24):**
   Do finishing tweaks and do project demonstration on 04/22/24 in lab.
5. **Week 16 (04/29/24):**
   Do individual group reflection, and submit final report.

---

## 9 List of Materials Expected

**Display Arduino**

1. 1x Arduino UNO-R3
2. 1x 16x2 LCD
3. 1x Nokia 5110 LCD
4. 1x Breadboard
5. 2x 10KΩ potentiometer
6. 4x 10KΩ resistors
7. 1x 220Ω resistor
8. 1x 330Ω resistor
9. 1x 1KΩ resistor

**Photoresistor Array Arduino**

1. 1x Arduino UNO-R3
2. 1x Breadboard
3. 4x 10KΩ resistor
4. 4x LEDs
5. 4x 220Ω resistors
6. 7x photoresistors.
7. 1x 8to1 multiplexor.

**Miscellaneous**

1. C++23 Compiler
2. Windows Computer
3. Lots of jumper wires.

---

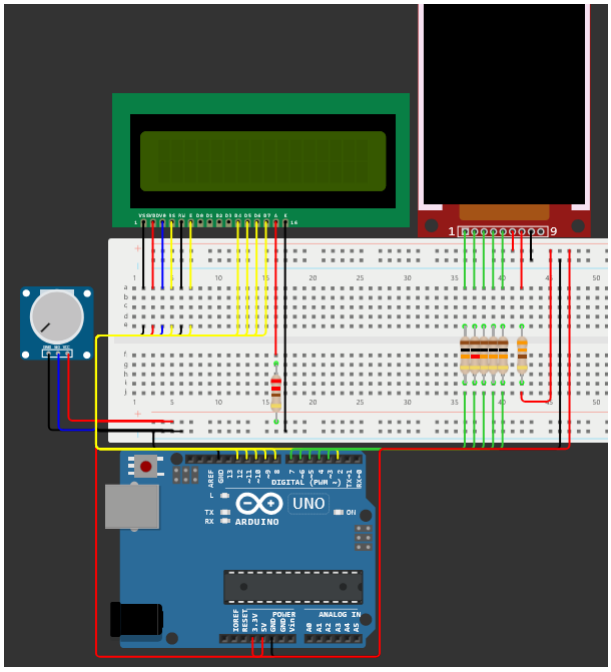
## 10 List of References

1. https://en.cppreference.com/w/cpp/header/chrono
2. https://en.cppreference.com/w/cpp/header/thread
3. https://learn.microsoft.com/en-us/windows/win32/api/winbase/ns-winbase-dcb
4. https://www.theengineeringprojects.com/2019/06/introduction-to-pn2222.html
5. https://www.youtube.com/watch?v=haUWO7BGYTE
6. https://www.build-electronic-circuits.com/rectifier-diode/
7. https://users.ece.utexas.edu/ valvano/Datasheets/PN2222-D.pdf
8. https://www.vishay.com/docs/88503/1n4001.pdf
9. https://www.onsemi.com/pdf/datasheet/tip120-d.pdf

---

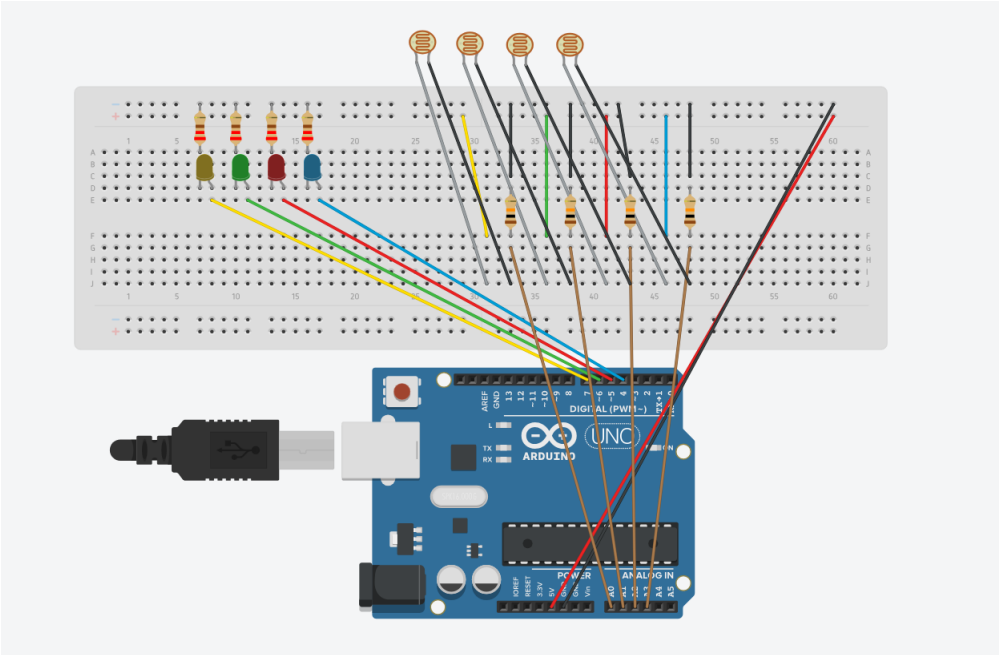
## 11 Diagrams + Pseudocode

Keystrokes and KPS Arduino:



---

Photoresistor Array Arduino:



---

Keystrokes and KPS Arduino:

1. Setup pins for both displays.
2. Wait for start signal from RX pin to start.
3. Loop until terminate signal read from RX.
   (a) Check if serial has data, read a short from it.
   (b) Using bitwise operators, get the KPS from the first 8 bits, and get the key presses from the last 8 bits.
   (c) Update displays with received data.

---

Photoresistor Array Arduino (WIP, only 4 photoresistors for 4K):

```
const uint8_t  pinValues[] = {A0, A1, A2, A3, 7, 6, 5, 4};
const uint16_t luxValues[] = {645, 665, 688, 597};
uint32_t prevTime = 0, curTime;
uint8_t  columnHits;


void setup()
{
    Serial.begin(9600);
    for (uint8_t i = 4; i < 8; ++i) {
        pinMode(pinValues[i], OUTPUT);
    }
}


void loop()
{
    curTime = millis();
    if (curTime - prevTime < 20) { return; }
    prevTime = curTime;

    columnHits = 0;
    for (uint8_t i = 0; i < 4; ++i)
    {
        if (analogRead(pinValues[i]) >= luxValues[i])
        {
            columnHits |= 1 << (3-i);
            digitalWrite(pinValues[i+4], HIGH);
        }
        else {
            digitalWrite(pinValues[i+4], LOW);
        }
    }
    Serial.print(columnHits ? columnHits : 16);
}
```

---