CS 362: Milestone 3

Due on March 5, 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 solenoid arrays, 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 solenoid 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. On the computer side, the program will read the hit timings from the .osu files and organize them into a hit-chart container. It will then gradually transmit this data to the solenoid array Arduino for physical key presses. We're also considering utilizing the Arduino's flash memory to store this data, although this aspect is still under consideration. The solenoid array Arduino will receive the .osu data and trigger the solenoids to execute physical key presses. The computer program will 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 receive hit press data from a computer-side program and activate corresponding solenoid keys. 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 solenoid Arduino to the display Arduino, 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 solenoid Arduino will receive beatmap 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 solenoid 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.

Expected Outputs

1. **Solenoid Activation:** The solenoid Arduino would output signals to activate the solenoids corresponding to the hit data received from the computer-side program. This would physically press keys on the keyboard.
2. **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.
3. **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.

---

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 solenoid 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, solenoid 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.

---

6 Timeline of Development

1. **Week 9 (03/11/24):**
   Ryan will be focused on finishing up the display Arduino code, and will be finishing up the computer-side program for the display Arduino.
   John's circuit catches on fire after connecting a 950 mA solenoid with a 600 mA transistor.
2. **Week 10 (03/18/24):**
   Ryan is helping John with the solenoid computer-side code.
   John is researching and prototyping solenoid Arduino.
3. **Week 11 (03/25/24):**
   Ryan will try making a GUI, otherwise will help John.
   John will try finishing up the software side for the solenoid.
4. **Week 12 (04/01/24):**
   Ryan will try making a GUI, otherwise will help John.
   John will try finishing up the software side for the solenoid.
5. **Week 13 (04/08/24):**
   Ryan will try making a GUI, otherwise will help John.
   John is finishing up solenoid Arduino.
   Make final stand for both Arduinos to sit on.
6. **Week 14 (04/15/24):**
   Create presentation slides and present on 04/15/24 in lab.
7. **Week 15 (04/22/24):**
   Do finishing tweaks and do project demonstration on 04/22/24 in lab.
8. **Week 16 (04/29/24):**
   Do individual group reflection, and submit final report.

---

7 List of Materials Expected

Display Arduino

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

Solenoid Array Arduino

1. 1x Arduino UNO-R3
2. 1x 1KΩ resistor
3. 4x 1n4007 Diode Rectifier
4. 4x NPN TIP122 Transistors
5. 4x BM-0530B Solenoid
6. 12V Barrel-Jack Power Supply

Miscellaneous

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

---

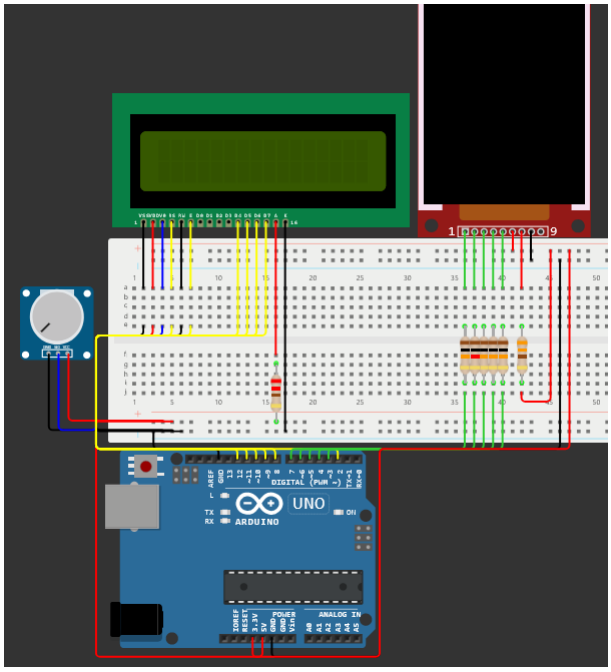
8 List of References

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

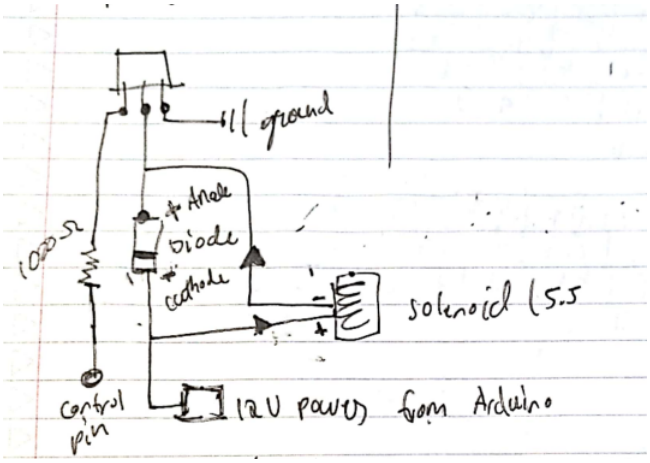
---

9 Diagrams + Pseudocode

Keystrokes and KPS Arduino:



Solenoid 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.

Solenoid Array Arduino:

1. Setup pins for solenoids.
2. Wait for start signal from RX pin to start.
3. Loop until terminate signal read from RX.
   (a) Read in hit press data when available.
   (b) Send correct signals to solenoids depending on next hit object in queue.
