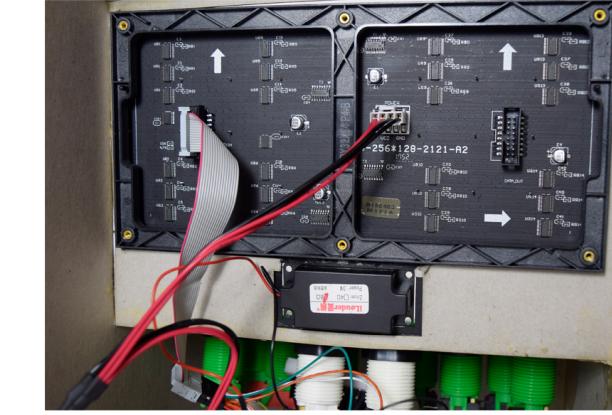
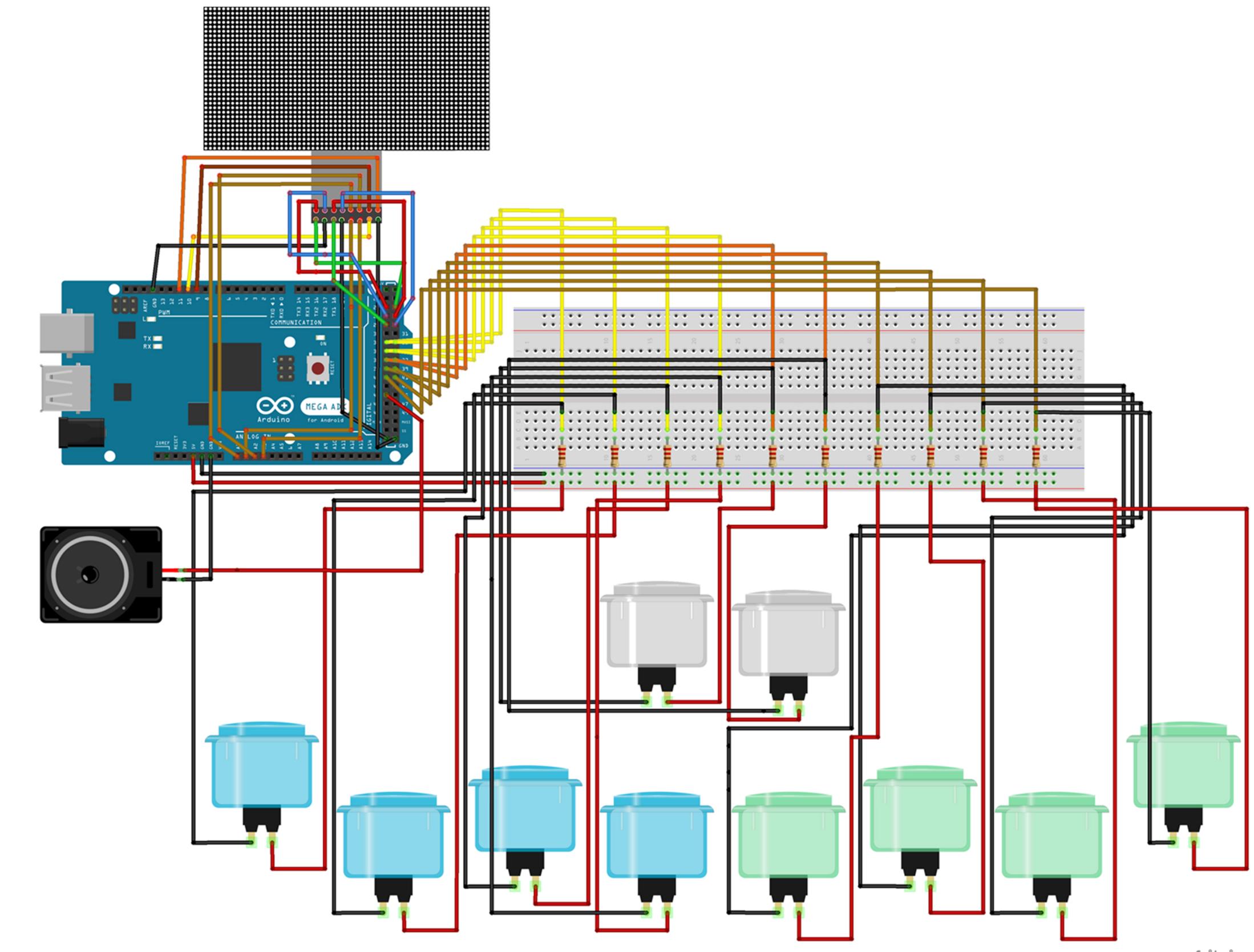


2-PLAYER TETRIS ARCADE MACHINE

BECK RIVERA



The 2-Player Tetris Arcade Machine is a project which uses the 64x32 RGB LED Matrix to run a multiplayer version of the classic arcade game Tetris. The machine is powered by an Arduino ATmega 2560. This is based off of another project, LX' Arduino Tetris by RomanSixty. The original machine was a single-player Tetris machine using the 32x32 LED Matrix. My additions include: a functional two-player system, a start screen to introduce players to the game, the Tetris theme, which plays through a speaker connected to the system, and a casing to hold the machine. Some of the original project's code was also modified to fix errors and to generally clean it up. The buttons were first daisychained together, but that led to occasional false positives which messed with gameplay, so now they're all connected individually. The song itself is also mutable. While in the start screen, press either the P1 or P2 Rotate buttons to mute the song, and press each P1 or P2's Fast Drop button to unmute it. The song only plays during gameplay, and not in the menu.



```
void loop() {
    //small buffer for program to run correctly
    Serial.println(noteDelayCounter);

    if(game_over && game_overP2) {
        if(countdownTimer >= 1500) {
            // fill the screen with 'black'
            matrix.fillRect(0, 0, 64, 32, matrix.Color333(0, 0, 0));
        }
        // draw some text!
        matrix.setCursor(1, 0);
        matrix.setTextSize(1); // size 1 == 8 pixels high
        matrix.setTextWrap(false); // Don't wrap at end of Line - will do ourselves

        matrix.setTextColor(matrix.Color333(7, 4, 0));
        matrix.println();
        matrix.print(" TETRIS");

        matrix.setTextSize(0.5);
        matrix.setTextColor(matrix.Color333(7,7,7));
        matrix.println("Press Start!");

        countdownTimer = 0;
    } else {
        countdownTimer++;
    }
    if ((digitalRead(p1_Ro) == HIGH) || (digitalRead(p2_Ro) == HIGH)) {
        mute = true;
    }
    else if ((digitalRead(p1_D) == HIGH) || (digitalRead(p2_D) == HIGH)) {
        mute = false;
    }
}

if(!game_over || !game_overP2) && (mute == false) {
    if(noteDelayCounter == 0) {
        curNoteDuration = baseNoteDuration/noteDurations[thisNote];
        tone(speakerPin, melody[thisNote], curNoteDuration);
        pauseBetweenNotes = curNoteDuration / 2.3;
        noteDelayCounter++;
    }
}
```