ArduRUNo!

A fun LCD game powered by an Arduino controller board

THE TECHNICAL ACE

Technology shall be aced

TheTechnicalAce.com

By: Jaryn D. Giampaoli

Merced College – Spring 2020

Computer Architecture & Organization

CPSC-42-1191

Table of Contents

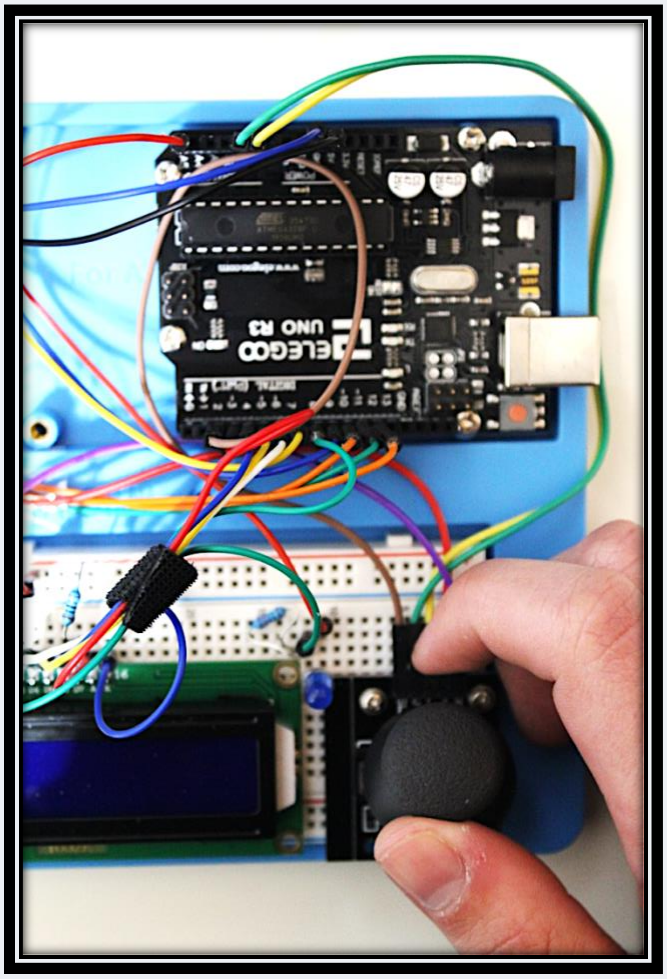
SYNOPSIS

Utilizing code developed by Mohammed Magdy from the Arduino Project Gallery that only provided the graphics and just utilized a boring switch for gameplay, ArduRUNo is a deeply enhanced spinoff of Magdy's work.

This document emphasizes on how game levels, cool light effects with LEDs, and awesome melodies and/or tones were combined to the base code of Magdy to create a revolutionary and more entertaining game powered by an Arduino microcontroller.

Key Features

- Game High Score: Keeps track of the highest score to ever be obtained in the game and makes a player want to continue playing to try and beat it.
- Level Increases: During gameplay, the game will automatically change to a faster speed in intervals of 50 (uses the player's current score in-game).
- Usage of LEDs: Makes the game more entertaining and acts as an additional guide for a player to know what is occurring or has occurred.
- Sound: With the implementation of a passive piezoelectric buzzer, the game becomes more fun! From a signature game start-up melody, to a "3,2,1, go!" countdown before the game starts, along with a short melody that is played each time the player levels up, this is what truly makes a game fun.
  - **BONUS:** *A melody to the tune of "The Final Countdown" by Europe is played once a player reaches the IMPOSSIBLE level!*



The Whole is More than the Sum of its Parts

-Aristotle

PARTS UTILIZED

The following parts were used to accomplish this task (* means optional):

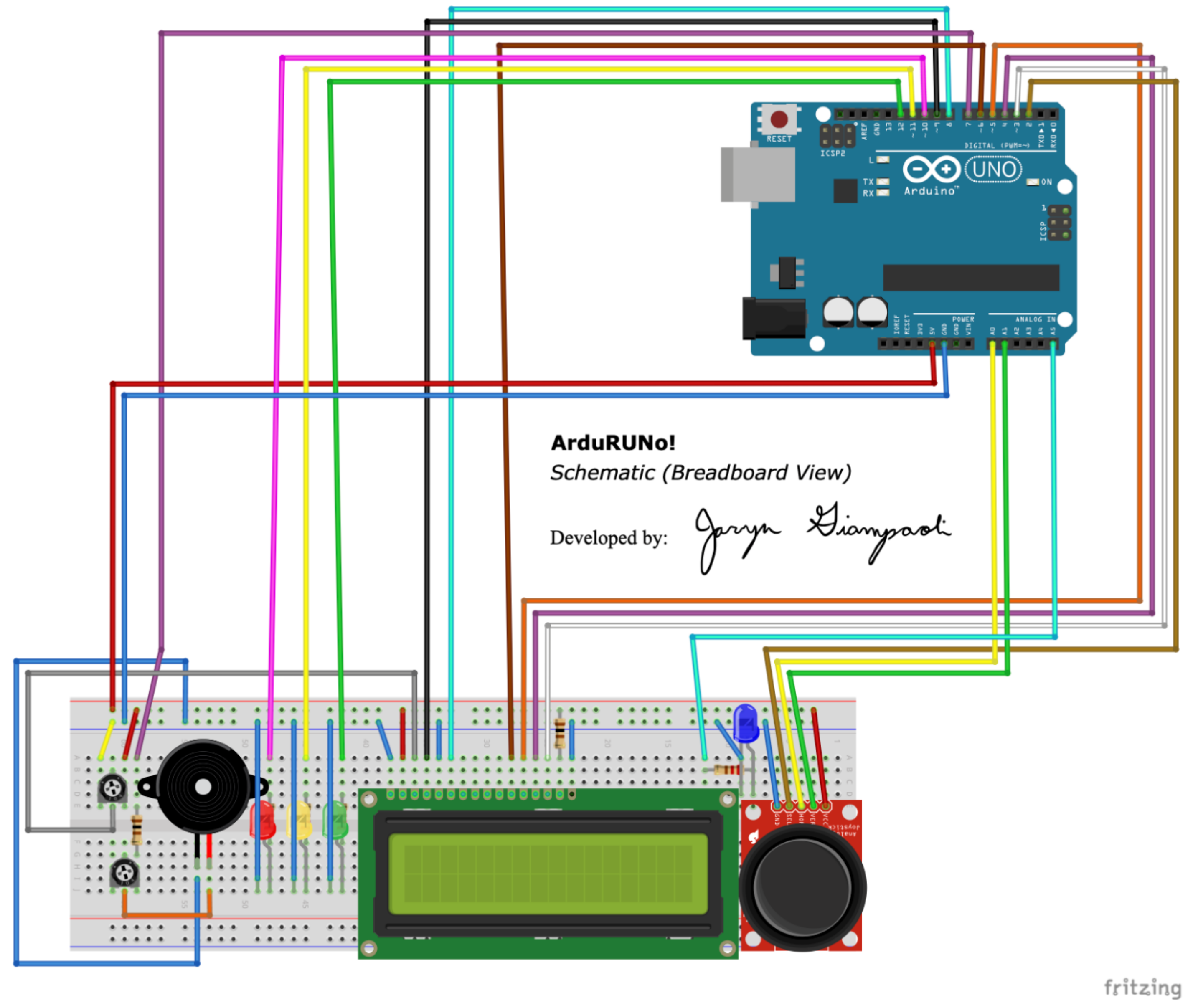
- *1 of UNIROI Mounting Board for Arduino and/or Raspberry Pi
- 1 of Elegoo Uno R3 Controller Board
- 1 of 16x2 LCD Display w/ Blue Backlight (Model#: 1602A)
- 1 of Analog Joystick
- 1 of Passive Piezoelectric Buzzer
- 1 of DiCuno 5mm (Round Head) Blue LED
- 1 of DiCuno 5mm (Round Head) Red LED
- 1 of DiCuno 5mm (Round Head) Yellow LED
- 1 of DiCuno 5mm (Round Head) Green LED
- 2 of 10K Potentiometers
- 4 of 220Ω Resistor
- 2 of 100Ω Resistor
- 26 of Breadboard Jumper Wire
- 5 of Female-to-Male Dupont Wire

Specifications of LEDs Used

●	●	●	●
BLUE	**RED**	**YELLOW**	**GREEN**
20mA	20mA	20mA	20mA
3.0-3.2v	2.0-2.2v	2.0-2.2v	3.0-3.2v
460-465nm	620-625nm	590-595nm	520-525nm

Project Circuit Construction

The following is a schematic in breadboard style view. If you would prefer the schematic in an engineering view, please view the information listed below the following breadboard style schematic. Be sure to obtain the proper parts and reverify the amount of resistance for the resistors to ensure an improper one is not used.



Scan the QR code to the right of this text to view and/or download the engineering style of the diagram:

Underlying Code of the Project

For this report, you will find algorithms of the most significance to the overall code of the project. All algorithms mentioned in this report were originally created by Jaryn D. Giampaoli. The term "MN" signifies the method name for the algorithm (if declared as such) within the overall code. To view the project's code in its entirety, seek the QR code at the bottom of page ten.

Game High Score

Despite this algorithm not being very complex to code, it is crucial to the overall functionality of the ArduRUNo game.

- **Figure 1** showcases the code that is used to compare the player's current score (inGameScore) to the last recorded high score. If the player's current score is greater than the last recorded high score, the new high score becomes that player's score and a static boolean variable named "highScoreAchieved" changes to true to later signal what to do when a new high score is achieved.

- **Figure 2** showcases the code that works in tandem to *Figure 1*. As the static boolean variable named "highScoreAchieved" is initialized as false, this code will only execute if "highScoreAchieved" returns a true value. This code flashes the blue LED rapidly four times and also displays a notice for 6.5 seconds to the player through the LCD indicating that a new high score has been achieved. A delay for a length of 1.5 seconds is declared at the very start to ensure a smooth transition from the tasks executed when game ends to when a new high score is achieved.

To view the aforementioned figures, seek the appendices page.

Implementing an Analog Joystick

Originally, this game was not very fun functionality wise. It only consisted of a simple push button that would have to be pressed each time to make the character within the game jump to avoid an obstacle. It was also observed that the push button sometimes was not very responsive to when the player needs to have that task executed rapidly. After researching on how to wire and implement a joystick with the Arduino controller board, the base code was analyzed and modified slightly. In the setup loop (not shown), code is written to record the positions of the x-axis and y-axis of the joystick. Listed below shows my simple modification to the base code. All that was needed was to change the parameter for the *IF* loop.

```
if (xPosition <= 20 || yPosition <= 300) {
    if (RUNNING_MANPos <= RUNNING_MAN_POSITION_RUN_LOWER_2)
        RUNNING_MANPos = RUNNING_MAN_POSITION_JUMP_1;
    joystickPressed = false;
  }
```

Game Levels (MN: gameSpeed)

This algorithm is responsible for controlling the speed of the game in intervals of 50 on the player's score. During gameplay, this code runs continuously and the delay value (otherwise known as the game speed) will decrease if the player's score falls into a specified interval. Additionally, a corresponding method is called for each interval to signify to the player that the game has increased in difficulty.

```
/**GAME LEVEL INCREASE OF SPEED SETTINGS*/
void gameSpeed() {
  if (gameLevelIndicator < 50) {
    delay(105);//EASY
  } else if ((50 < gameLevelIndicator) && (gameLevelIndicator < 100)) {
    level1Tone();
    delay(100);//MEDIUM
  } else if ((100 < gameLevelIndicator) && (gameLevelIndicator < 150)) {
    level2Tone();
    delay(95);//HARD
  } else if ((150 < gameLevelIndicator) && (gameLevelIndicator < 200)) {
    level3Tone();
    delay(90);//EXPERT
  } else if ((200 < gameLevelIndicator) && (gameLevelIndicator < 250)) {
    level4Tone();
    delay(80);//MASTER
  } else if ((250 < gameLevelIndicator) && (gameLevelIndicator < 300)) {
    level5Tone();
    delay(75);//PROFESSIONAL
  } else if (300 < gameLevelIndicator) {
    levelMasterTone();
    delay(70);//IMPOSSIBLE
  } else {
    delay(105);//Give program time to shift to new gameSpeed
  }
}//End gameSpeed
```

LED Stoplight Game Start Countdown Timer

With this algorithm, the game immediately begins to feel intense. The LCD is programmed to display "GET READY…" on the first row, with the second row displaying "In 3…", "In 2…", In 1…" in an increment of one second respectively. Additionally, when on 3, the red LED light flashes and a tone with a low pitch is played. When on 2, the yellow LED light flashes and a tone with a mid-pitch is played. Lastly, when on 3, the green LED light flashes and a tone with a high pitch is played for a slightly longer duration. After this piece of code is executed, the game begins!

To view the corresponding code, seek the appendices page.

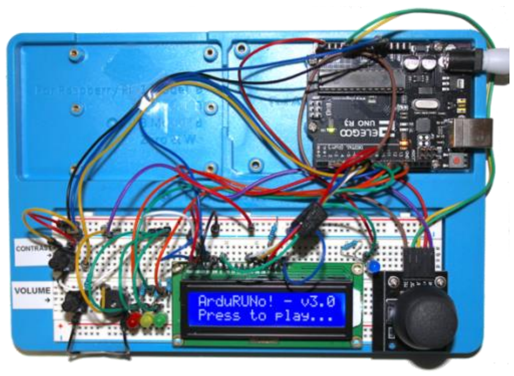
Continue to Next Page →

Melody for Impossible Level (MN: levelMasterTone)

This algorithm is responsible for playing the melody to the tune of "The Final Countdown" by Europe. This melody will only play one time once a player reaches the *IMPOSSIBLE* (highest) level. Delays **are not** utilized within this algorithm. This approach had to be utilized as this method is called within the gameSpeed method which already has delays. As the delays contained within the gameSpeed method control the speed of the game, they **cannot** be changed or be in parallel with other delays.

Scan the QR code to the right of this text to view this project's code in all of its entirety:

Physical Appearance of Project



The physical appearance and functionality of this game was given priority over the actual underlying code for this project. As I do many web and graphics design, I aim to keep a clean design in both hardware and software. Thus, the layout of the hardware components utilized were strategically placed to ensure a clean design.

Challenges Endured

Overall, this project was extremely fun to construct, but there were times where my original modifications would not play in harmony with the base code provided by Magdy. Hours of research permitted myself to try different alternatives to eventually obtain a solution. The main difficulty endured was finding a way to make the passive piezoelectric buzzer sound while in gameplay as I could not utilize delays as it would cause the overall functionality of the game to break. Eventually I recalled that the "blink without delay" function does relatively the same thing, so using a modification of that example led to myself solving the error presented.

Works Cited

Arduino. (2015, July 28). *Blink Without Delay*. Retrieved from Arduino: https://www.arduino.cc/en/tutorial/BlinkWithoutDelay

Magdy, M. (2016, October 18). *Arduino Game By LCD*. Retrieved from Arduino Project Hub: https://create.arduino.cc/projecthub/muhamd-magdy/arduino-game-by-lcd-9a3bc2?ref=search&ref_id=lcd%20game&offset=5

Appendices

Corresponding Code for Game High Score – Figure 1

```
/*Keep track of the player's score to indicate
    when it's time to increase the current level
    of the game*/
  gameLevelIndicator = inGameScore;

  /*Overwrites previous game high score if
    player's score has broken the record*/
  if (inGameScore > gameHighScore) {
    gameHighScore = inGameScore;
    highScoreAchieved = true;
  }
```

Corresponding Code for Game High Score – Figure 2

```
/**HIGH SCORE FUNCTIONALITY*/
    if (highScoreAchieved) {
      delay(1500);

      gameHighScoreBuzzer();

      digitalWrite(ledColorsByPin[0], HIGH);
      delay(delayOfHighScoreLED);
      digitalWrite(ledColorsByPin[0], LOW);   // turn the LED off
      delay(delayOfHighScoreLED);
      digitalWrite(ledColorsByPin[0], HIGH);
      delay(delayOfHighScoreLED);
      digitalWrite(ledColorsByPin[0], LOW);   // turn the LED off
      delay(delayOfHighScoreLED);
      digitalWrite(ledColorsByPin[0], HIGH);
      delay(delayOfHighScoreLED);
      digitalWrite(ledColorsByPin[0], LOW);   // turn the LED off
      delay(delayOfHighScoreLED);
      digitalWrite(ledColorsByPin[0], HIGH);
      delay(delayOfHighScoreLED);
      digitalWrite(ledColorsByPin[0], LOW);   // turn the LED off
      delay(delayOfHighScoreLED);

      lcd.setCursor(0, 0);
      lcd.println("NEW HIGH SCORE  ");
      lcd.setCursor(0, 1);
      lcd.print(gameHighScore);
      lcd.print("                ");
      highScoreAchieved = false;
      delay(6500);//Keep high score displayed for 6.5 secs before returning to title screen
    }
```

Continue to Next Page →

Corresponding Code for Game Start Countdown Timer

```
/**3 SECOND LED STOPLIGHT COUNTDOWN FOR GAME START**/
      lcd.setCursor(0, 0);
      lcd.print("   GET READY!   ");

      if ((switchValue == LOW) && (gameStartLEDtimesBlinked <= 3)) {
        //On your marks!
        digitalWrite(ledColorsByPin[1], HIGH);//Turn the RED LED on
        lcd.setCursor(0, 1);
        lcd.print("     In 3...    ");//Display instructions to play game
        tone(7, 1109, 300);
        delay(1000);// Keep LED on for 1 second

        digitalWrite(ledColorsByPin[1], LOW);   // turn the LED off
        delay(300);              // wait slightly to transition

        //Get set!
        digitalWrite(ledColorsByPin[2], HIGH);//Turn the YELLOW LED on
        lcd.setCursor(0, 1);
        lcd.print("     In 2...    ");//Display instructions to play game
        tone(7, 2093, 300);
        delay(1000);//Keep LED on for 1 second

        digitalWrite(ledColorsByPin[2], LOW);   // turn the LED off
        delay(300);              // wait slightly to transition

        //GO!
        digitalWrite(ledColorsByPin[3], HIGH);   //Turn the GREEN LED on
        lcd.setCursor(0, 1);
        lcd.print("     In 1...    ");//Display instructions to play game
        tone(7, 3136, 800);
        delay(1000);              // wait for a second
        digitalWrite(ledColorsByPin[3], LOW);   // turn the LED off
        delay(300);              // wait slightly to transition
        gameStartLEDtimesBlinked++;
      }
```

Continue to Next Page →

Corresponding Code for Melody for Impossible Level

```
void levelMasterTone() {
  currentMillis = millis();
  unsigned long noteDuration = 1000 / theFinalCountdownDuration[levelMasterNotesOutputted];
  pauseBetweenNotes = noteDuration * 0.3;

  if (levelMasterNotesOutputted < levelMasterNumberOfNotes) {
    if (outputTone) {
      if (currentMillis - previousMillis >= noteDuration) {
        previousMillis = currentMillis;
        noTone(buzzer);
        outputTone = false;
        levelMasterNotesOutputted++;
      }
    }

    else {
      if (currentMillis - previousMillis >= pauseBetweenNotes / 4) {
        previousMillis = currentMillis;
        if (theFinalCountdownMelody[levelMasterNotesOutputted] == 0) {
          noTone(buzzer);
        } else {
          tone(buzzer, theFinalCountdownMelody[levelMasterNotesOutputted]);
        }
        outputTone = true;
      }
    }
  }
}
```

ArduRUNo!
Current Version: 3.0
Developed by: Jaryn D. Giampaoli

THE TECHNICAL ACE
Technology shall be aced