Robotic Drink Mixer Powered by Arduino (December 2017)

Emmanuel Joseph R. Talusan, *Member, IEEE*, and Huayang Zhang, *Member, IEEE*
Advisor: Kuo-Sheng Ma
Department of Electrical Engineering and Computer Science (EECS)
LOYOLA MARYMOUNT UNIVERSITY
Los Angeles CA, USA

***Abstract*—Arduino is an open-source microcontroller that can be designed into different electronic devices. In this research project, Arduino is used to create an automated drink mixer. This machine integrates different input and output components to maximize the microcontroller's functionality. The design of the machine bases on the safety and simplicity of the device. The user-interface was designed in the principle of minimizing the effort from the user. Once the user selected the drink order, the rest of the steps will be fully automated.**

I. Introduction

Among the robotic drink mixer in the market, most of the machines focus on selling to the luxury business. However, there is a niche market of selling to regular household cocktail drinkers. And that market has the potential customers for this robot. The design of the machine is based on simplicity and consumer friendliness.

The simplicity in design can be realized through the mass amount of resources supplied through the Arduino community. Arduino provides an extensive open-source code base as well as economical add-on components. To be illustrate the system, Figure 1 shows the state diagram of the program.

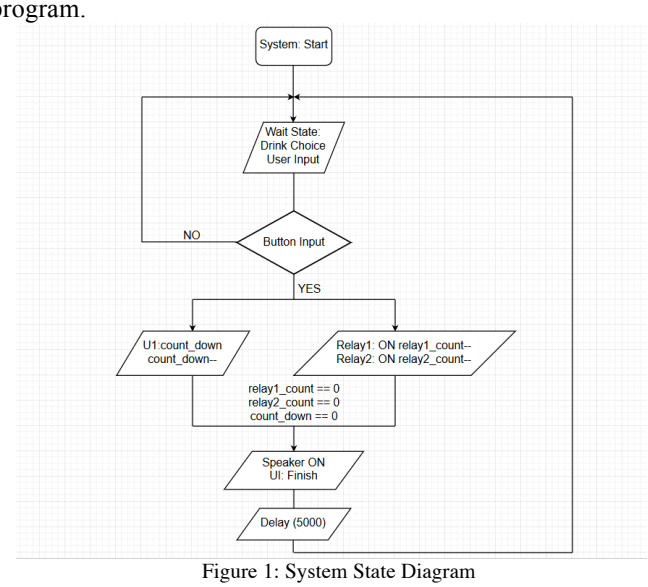

Figure 1: System State Diagram

There are mainly three states within the system. 1. Waiting State; 2. Pouring State; 3. Finish State. State 1 transition into state based solely on user input. Transition from State 2 to State 3 bases on a counter counting down to 0. The counter corresponding to how long it takes for the robot to pour the drink. Once the counter reaches 0, the machine will transition into the State 3, which is to alert the user that the drink is ready. After staying State 3 for 5 seconds, the machine transitions back into State 1 and waiting for the user's next input. The entire software part of the system is being processed on the Arduino microcontroller.

II. Problem Statement

An affordable robotic drink mixer can provide tremendous convenience to regular consumers. However, there is yet to be an economical and full functional model out in the market. Some additional problems that were addressed including finding different I/O devices work well with Arduino, using Arduino to switch on and off electrical devices, 3D printing device shell, the feasible voltage and current for peristaltic pumps.

III. Proposed Solution

The system is designed with an Arduino microcontroller as the centralized processing unit. This product utilize non-proprietary software, 3D printed shell, and economical hardware components. The external I/O components are listed as follows:

- 1 – Membrane 1x4 Keypad
- 1 – 1x4 Channel 240V 10A Relay Module Board
- 2 – Peristaltic Liquid Pump with Silicone Tubing
- 1 – 12V 5A switching power supply
- 1 – Female DC Power adapter - 2.1mm jack to screw terminal block
- 1 – Adafruit 1.44" Color TFT LCD display
- 1 – Piezo Buzzer - PS1240

The Arduino system is being powered by 5V output from a USB port. The 12V 5A switching power supply the power for the peristaltic pumps. The system initially interacts with the user through the Adafruit 1.44" Color TFT LCD display. The connection between the display and Arduino is shown in Figure2.

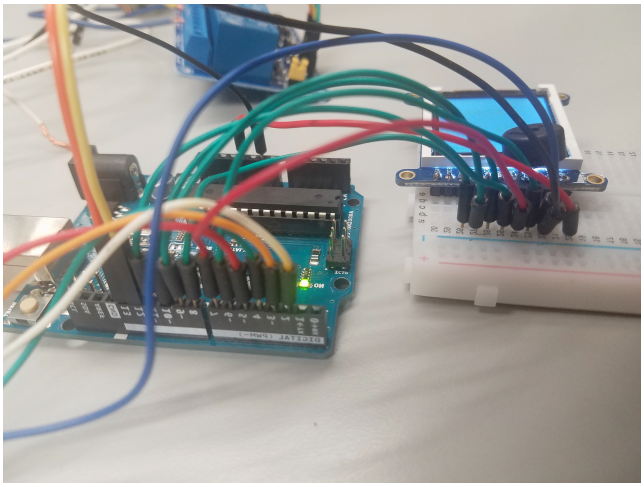

Figure 2: Arduino to LCD Pin Connection

The user is prompted for three drink choices as shown in Figure 5 in the attachment. User select the desired drink choice using the 1x4 membrane keypad. The Arduino microcontroller then process the input and start pouring the drink for the user. Pouring is triggered through Arduino sending a high signal to the relay, which is in series connection with the peristaltic pumps. As the relay stays connected, the pump will keep pumping drinks from the ingredient cups to the output cup. As the drink being pumped, the LCD displays a countdown as shown in Figure 6 in the attachment. Once the pumps finish pumping, the Arduino will activate the Piezo Buzzer to play a simple tune to alert the user. The display also shows the user when the drink is ready, as illustrated in Figure 7 in attachment.

IV. Experimental Results

The drink mixer system was a success. According to the plan, the system starts with the Waiting State, Pouring State, Finish State, and then back to the Waiting State. The I/O components mostly work as desired. However, the Membrane 1x4 Keypad malfunctions from time to time. The hypothesis for the error is either the product was flawed or there need to be resistors connected in series for the buttons to pull the voltage down properly. The buzzer play the tune pitch perfectly as desired. The rate of pumping liquid is consistent. Each pump was enabled and disabled based on the state of the relay, and the relay worked well with the state of the Arduino. Figure 8 in the attachment shows all the electrical components used in the project.

This minimum viable product meets the goal that set up for this project. There are some improvements can be implemented in the future to increase this machine's utility. A casing such as shown in Figure 3 can assist in organizing components for the system.

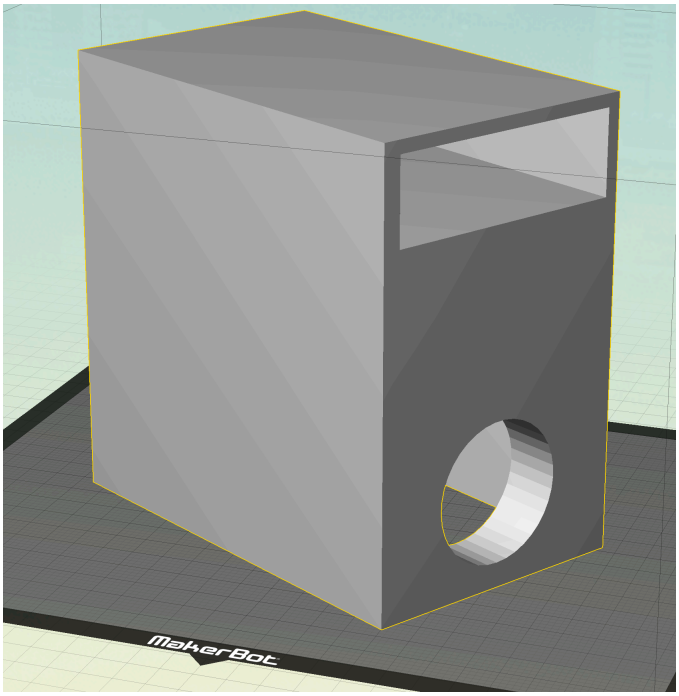

Figure 3: 3D Modeled Shell

A shell like this not only helps the organization, but it also gives this product a very nice presentation.

From researching similar products on the market, a product like this should offer wireless capability, using either Arduino wifi module or the Arduino Bluetooth shield. To better illustrate this possibility, a mockup smartphone app screen is designed as shown in Figure 4.

Figure 4. Mockup iPhone User Interface

V. Conclusion

The project involves different components with the Arduino. Each components were tested individually and then was put together with the Arduino as a whole. This modular approach ensure the safety of the project. The project was able to address to problem that there is no affordable and functional drink mixer in the market. The proposed solution is being proven by this product. So even though there are still quite improvements need to be done, but overall, this project is a success.

References

[1] Kinsman, Ted. "Build a Simple Cocktail Drinkbot with Arduino | Make:" *Make: DIY Projects and Ideas for Makers*, makezine.com/projects/build-cocktail-drinkbot/.

Attachment:

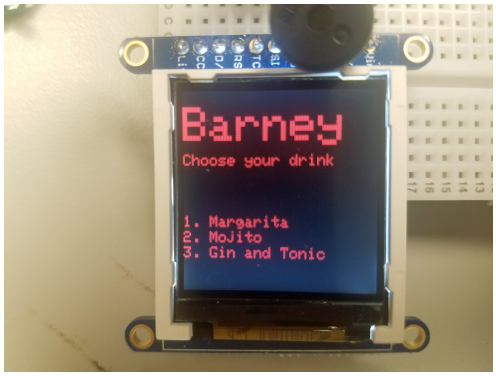

Figure 5. Wait State Screen

Figure 6. Pouring State Screen

Figure 7. Finish State Screen

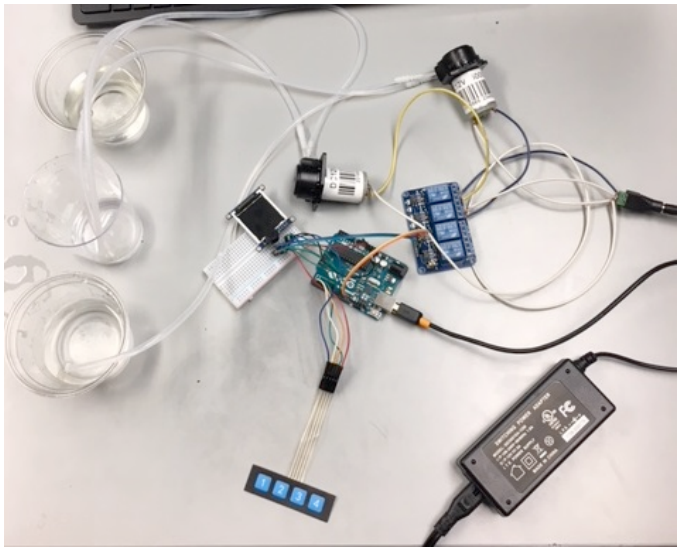

Figure 8. Electrical Components of the System

Arduino Code:

```
#include <Adafruit_GFX.h>    // Core graphics library
#include <Adafruit_ST7735.h> // Hardware-specific library
#include <SPI.h>
#include <math.h>

// Setup frequency for the music notes
#define NOTE_G3  196
#define NOTE_A3  220
#define NOTE_B3  247
#define NOTE_C4  262

// Setup the display
#define TFT_CS     10
#define TFT_RST    9  // you can also connect this to the Arduino reset
                      // in which case, set this #define pin to -1!
```

```
#define TFT_DC     8
#define TFT_SCLK   6   // set these to be whatever
pins you like!
#define TFT_MOSI   11   // set these to be
whatever pins you like!
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS,
TFT_DC, TFT_MOSI, TFT_SCLK, TFT_RST);


// notes in the melody:
int melody[] = {
 NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3,
NOTE_G3, 0, NOTE_B3, NOTE_C4
};

// set up output pins for the pump
int pinOut1 = 13;
int pinOut2 = 12;

// note durations: 4 = quarter note, 8 = eighth note,
etc.:
int noteDurations[] = {
 4, 8, 8, 4, 4, 4, 4, 4
};
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7
= 2;

// setup all the buttons
const int button1 = 3;
const int button2 = 2;
const int button3 = 5;
const int button4 = 4;
// all buttons set to LOW
int buttonState1 = 0;
int buttonState2 = 0;
int buttonState3 = 0;
int buttonState4 = 0;

// setup initial time count down values
int relay1_count = 10;
int relay2_count = 8;
int count_down = max(relay1_count, relay2_count);

void setup(void) {
 pinMode(button1, INPUT);
 pinMode(button2, INPUT);
 pinMode(button3, INPUT);
 pinMode(button4, INPUT);

// set up water pump pins
 pinMode(13, OUTPUT);
 pinMode(12, OUTPUT);
 Serial.begin(9600);

// setup display
 tft.initR(INITR_144GREENTAB);   // initialize a
ST7735S chip, black tab
}
int flip = 0;


void loop() {
 digitalWrite(pinOut1, HIGH);
 digitalWrite(pinOut2, HIGH);
 buttonState1 = digitalRead(button1);
 buttonState2 = digitalRead(button2);
 buttonState3 = digitalRead(button3);
 buttonState4 = digitalRead(button4);

 uint16_t time = millis();
 tft.fillScreen(ST7735_BLACK);
 time = millis() - time;

 Serial.println(time, DEC);
 delay(500);

// large block of text
 tft.fillScreen(ST7735_BLACK);
 tftPrintTest();
 delay(4000);

 tft.invertDisplay(flip);
 if ((buttonState1 == HIGH) || (buttonState2 ==
HIGH) || (buttonState3 == HIGH) || (buttonState4
== HIGH)) {
   digitalWrite(pinOut1, LOW);
   digitalWrite(pinOut2, LOW);

   for (int i = count_down; i>=0; i--) {
     tft.setTextColor(ST7735_BLUE);
     tft.fillScreen(ST7735_BLACK);
     tft.setTextSize(2);
     tft.setCursor(0, 40);
     tft.println("Wait for");
     tft.setCursor(0, 80);
     tft.println(i);
     tft.setCursor(30, 80);
     tft.println("secs");
     delay(1000);
   }
   digitalWrite(pinOut2, HIGH);
   digitalWrite(pinOut2, HIGH);

   buttonState1 = LOW;
   buttonState2 = LOW;
   buttonState3 = LOW;
   buttonState4 = LOW;
```

```
   // new UI after pour the drink
   tft.setTextColor(ST7735_BLUE);
   tft.fillScreen(ST7735_BLACK);
   tft.setTextSize(3);
   tft.setCursor(0, 20);
   tft.println("Enjoy");
   tft.setCursor(0, 60);
   tft.println("Your");
   tft.setCursor(0, 100);
   tft.println("Drink");
   delay(2000);
   // music after finish the drink
   for (int thisNote = 0; thisNote < 8; thisNote++) {

     // to calculate the note duration, take one
second divided by the note type.
     //e.g. quarter note = 1000 / 4, eighth note =
1000/8, etc.
     int noteDuration = 1000 /
noteDurations[thisNote];
     tone(7, melody[thisNote], noteDuration);

     // to distinguish the notes, set a minimum time
between them.
     // the note's duration + 30% seems to work well:
     int pauseBetweenNotes = noteDuration * 1.30;
     delay(pauseBetweenNotes);
     // stop the tone playing:
     noTone(7);
   }
 }
 flip = 1 - flip;
 delay(2000);
}

void tftPrintTest() {
 tft.setTextWrap(false);
 tft.fillScreen(ST7735_BLACK);
 tft.setCursor(0, 10);
 tft.setTextColor(ST7735_RED);
 tft.setTextSize(3);
 tft.println("Barney");

 tft.setTextSize(1);
 tft.setCursor(0, 40);
 tft.print("Choose your drink");
 tft.setTextSize(1);
 tft.setCursor(0, 80);
 tft.print("1. Margarita");
 tft.setCursor(0, 90);
 tft.print("2. Mojito");
 tft.setCursor(0, 100);
 tft.print("3. Gin and Tonic");
}
```