Candy Sorter - Bivolaru Andra 333CA

Introducere

Proiectul meu constă în construirea unui sistem de sortare a bomboanelor utilizând platforma Arduino. Scopul acestuia este de a automatiza procesul de sortare a bomboanelor în funcție de culoare și de a afișa numărul de bomboane sortate și tipul de bomboană detectată pe un ecran LCD.
Am dezvoltat acest dispozitiv care oferă o soluție de sortare și aranjare a obiectelor mici dupa culoare, pornind de la necesitatea oamenilor de a-și organiza medicamentele pe care trebuie să le ia într-o zi.

Descriere generală

Automatul de sortare a bomboanelor utilizează urmatorul sistem pentru a procesa și distribui bomboanele în funcție de culoare:

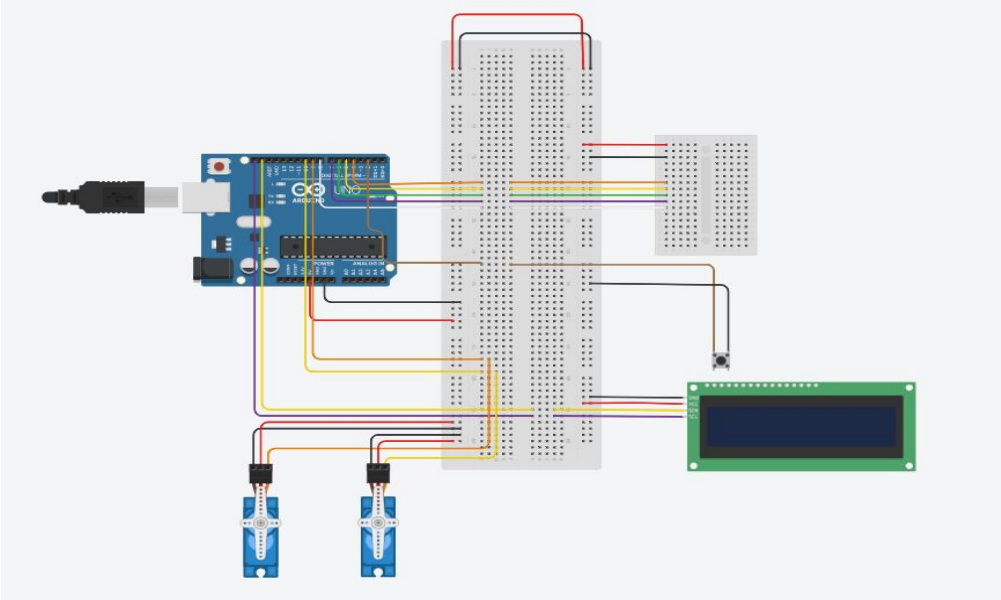
1. Un tub lanseaza pe rand bomboane pe un disc rotativ
2. Discul preia bomboana
3. Discul se roteste cu ajutorul unui servo motor pana ajunge la senzorul de culoare
4. Senzorul scaneaza bomboana si deduce culoarea acesteia
5. Pe ecranul LCD se afiseaza aroma bomboanei si numarul total de bomboane care au fost sortate
6. Pe urma, un alt motor servo dirijeaza o panta inspre paharul corespunzator culorii bomboanei
7. Discul se roteste si bomboana cade pe panta in pahar
8. Discul se roteste pana in punctul de start si preia urmatoarea bomboana

Hardware Design

Lista componentelor folosite în cadrul proiectului:

- Arduino UNO
- ECRAN LCD 1602 - CHIP AIP31066
- MOTOR SERVO SG90 9G
- TCS230 Color Sensor Module

Schema electrica:



Breadboard-ul mini din partea dreapta reprezinta Color Module-ul, deoarece nu exista pe Tinkercad; unde pinii 4, 5, 6, 7, 8 din Arduino sunt conectati la pinii S0, S1, S2, S3, OUT de pe TCS230.

Software Design

Codul Arduino:

```
/*
  Include the used libraries
*/
#include <Servo.h>
#include <SoftwareSerial.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

/*
  Defines
*/
#define DISCPIN 9
#define SLIDEPIN 10
#define BUTTONPIN 2
#define COLOR_S0 4
#define COLOR_S1 5
#define COLOR_S2 6
#define COLOR_S3 7
#define COLOR_OUT 8

/*
  The variables used to move the two servos
*/
Servo discServo, slideServo;
```

```
int moveDisc = 0;
int moveSlide = 0;

/*
  Variables for color sensor PW, Calibration Values
*/
int redMin = 22; // Red minimum value
int redMax = 76; // Red maximum value
int greenMin = 26; // Green minimum value
int greenMax = 100; // Green maximum value
int blueMin = 8; // Blue minimum value
int blueMax = 30; // Blue maximum value

// Variables for Color Pulse Width Measurements
int redPW = 0;
int greenPW = 0;
int bluePW = 0;

// Variables for final Color values
int redValue;
int greenValue;
int blueValue;

/*
  Set the LCD to display number and set the outputs
*/
LiquidCrystal_I2C lcd(0x27,16, 2);

const char *redColor = "RED - CHERRY";
const char *greenColor = "GREEN - LIME";
const char *purpleColor = "PURPLE - GRAPE";
const char *yellowColor = "YELLOW - LEMON";
const char *placeCandy = "Place Candy";
int numCandy = 0;

void setup() {
  // Setup Serial Monitor
  Serial.begin(9600);

  // Set the pins for the servos
  discServo.attach(DISCPIN);
  slideServo.attach(SLIDEPIN);

  // Set up the color sensor's pins
  // Set S0 - S3 as outputs
  pinMode(COLOR_S0, OUTPUT);
  pinMode(COLOR_S1, OUTPUT);
  pinMode(COLOR_S2, OUTPUT);
  pinMode(COLOR_S3, OUTPUT);

  // Set Sensor output as input
```

```
  pinMode(COLOR_OUT, INPUT);

  // Set Pulse Width scaling to 20%
  digitalWrite(COLOR_S0,HIGH);
  digitalWrite(COLOR_S1,LOW);

  // Set the lcd
  lcd.init();
  lcd.clear();
  lcd.backlight();

  // Set up the button conditions
  pinMode(BUTTONPIN, INPUT_PULLUP);
}

void loop() {
  // Set the servo variables to initial position
  moveDisc = 0;
  moveSlide = 90;

  moveDiscServo(moveDisc);   // Move disc server to initial position
  moveSlideServo(moveSlide); // Position the slide servo in the initial
position

  // Write on the lcd place candy
  writePlaceCandy();

  // Wait to place the candy
  while (digitalRead(BUTTONPIN) == HIGH) {
    // Wait until the button is pressed
  }

  // Clar the place candy output from the lcd
  lcd.clear();
  delay(100);

  // Slowly move disc server to color sensor
  for (int i = 0; i < 90; i++) {
    moveDisc = moveDisc + 1;
    moveDiscServo(moveDisc);
    delay(10);
  }
  delay(4000);

  // Deduce the color of the candy
  findColor();

  moveSlideServo(moveSlide);  // Move the slide under the correct cup
  numCandy = numCandy + 1;    // Increase the number of candies sorted
  writeOutput();              // Write on the LCD the type of candy and
number of candies
```

```
  // Move sensor to the slide
  for (int i = 0; i < 90; i++) {
    moveDisc = moveDisc + 1;
    moveDiscServo(moveDisc);
    delay(20);
  }
  delay(4000);
}


/*
  Functions to move the servo motors
*/
void moveDiscServo(int degrees) {
  int servoPosition = map(degrees, 0, 180, 0, 180);
  discServo.write(servoPosition);
}

void moveSlideServo(int degrees) {
  int servoPosition = map(degrees, 0, 180, 0, 180);
  slideServo.write(servoPosition);
}

/*
  Function to figure out the color of the candy
*/
void findColor() {
  int maxim = -1;
  int redFlag = 0;
  int greenFlag = 0;
  int blueFlag = 0;

  // Read Red value
  redPW = getRedPW();
  // Map to value from 0-255
  redValue = map(redPW, redMin,redMax,255,0);
  // Delay to stabilize sensor
  delay(200);

  // Read Green value
  greenPW = getGreenPW();
  // Map to value from 0-255
  greenValue = map(greenPW, greenMin,greenMax,255,0);
  // Delay to stabilize sensor
  delay(200);

  // Read Blue value
  bluePW = getBluePW();
  // Map to value from 0-255
  blueValue = map(bluePW, blueMin,blueMax,255,0);
```

```
  // Delay to stabilize sensor
  delay(200);

  // Print output to Serial Monitor
  Serial.print("Red = ");
  Serial.print(redValue);
  Serial.print(" - Green = ");
  Serial.print(greenValue);

  Serial.print(" - Blue = ");
  Serial.println(blueValue);

  int red = redValue;
  int green = greenValue;
  int blue = blueValue;

  maxim = red;
  if (maxim < green) {
    maxim = green;
  }

  if(maxim < blue) {
    maxim = blue;
  }

  if (red > 250 && green > 250 && blue > 250) {
    Serial.println("YELLOW");
    moveSlide = 63;
  } else {
    if(maxim == red) {
      Serial.println("RED");
      moveSlide = 90;
    } else {
      if (maxim == green) {
        Serial.println("GREEN");
        moveSlide = 120;
      } else {
        Serial.println("PURPLE");
        moveSlide = 60;
      }
    }
  }
}

// Function to read Red Pulse Widths
int getRedPW() {
  // Set sensor to read Red only
  digitalWrite(COLOR_S2,LOW);
  digitalWrite(COLOR_S3,LOW);
  // Define integer to represent Pulse Width
  int PW;
```

```
  // Read the output Pulse Width
  PW = pulseIn(COLOR_OUT, LOW);
  // Return the value
  return PW;
}

// Function to read Green Pulse Widths
int getGreenPW() {
  // Set sensor to read Green only
  digitalWrite(COLOR_S2,HIGH);
  digitalWrite(COLOR_S3,HIGH);
  // Define integer to represent Pulse Width
  int PW;
  // Read the output Pulse Width
  PW = pulseIn(COLOR_OUT, LOW);
  // Return the value
  return PW;
}

// Function to read Blue Pulse Widths
int getBluePW() {
  // Set sensor to read Blue only
  digitalWrite(COLOR_S2,LOW);
  digitalWrite(COLOR_S3,HIGH);
  // Define integer to represent Pulse Width
  int PW;
  // Read the output Pulse Width
  PW = pulseIn(COLOR_OUT, LOW);
  // Return the value
  return PW;
}

/*
  Function to write the output on the LCD
*/
void writePlaceCandy() {
  lcd.clear();
  lcd.setCursor(0, 0);

  for (int i = 0; i < strlen(placeCandy); i++) {
    lcd.setCursor(i, 0);
    lcd.print(placeCandy[i]);
  }
}

void writeOutput() {
  lcd.clear();   // Clear the previous output

  lcd.setCursor(0, 0);
  switch(moveSlide) {
    case 60:
```

```
      for (int i = 0; i < strlen(purpleColor); i++) {
          lcd.setCursor(i, 0);
          lcd.print(purpleColor[i]);
        }
      break;
    case 120:
      for (int i = 0; i < strlen(greenColor); i++) {
          lcd.setCursor(i, 0);
          lcd.print(greenColor[i]);
      }
      break;
    case 90:
      for (int i = 0; i < strlen(redColor); i++) {
          lcd.setCursor(i, 0);
          lcd.print(redColor[i]);
      }
      break;
    case 63:
      for (int i = 0; i < strlen(yellowColor); i++) {
          lcd.setCursor(i, 0);
          lcd.print(yellowColor[i]);
      }
      break;
  }

  // Print number of candy
  char buffer[16]; // Create a character buffer to store the converted
string
  itoa(numCandy, buffer, 10); // Convert the integer to a string
  const char* stringNumCandy = buffer;
  for (int i = 0; i < strlen(stringNumCandy); i++) {
    lcd.setCursor(i, 1);
    lcd.print(stringNumCandy[i]);
  }
}
```

Video proiect

Rezultate Obţinute

Concluzii

Acest proiect mi-a oferit o primă perspectivă asupra lucrului cu componente hardware și m-a făcut să înțeleg că, în ciuda faptului că software-ul poate fi perfect scris, comportamentul componentelor nu va fi mereu același la fiecare rulare.

Am observat acest aspect în cazul senzorului de culoare, pe care a trebuit să-l calibrez de trei ori pentru a obține o preluare corespunzătoare a culorilor, precum și în cazul ecranului LCD, pe care l-am schimbat de trei ori din motive de incompatibilitate cu Arduino (din diverse motive).

Această experiență m-a determinat să înțeleg că nu este suficient să știi doar să lucrezi cu codul, ci și să înțelegi componentele care rulează codul și să descoperi de ce uneori funcționează, iar alteori nu.

Proiectul mi-a plăcut foarte mult și, deși ideea în sine a fost complexă din punct de vedere hardware și software, am dorit să adaug și o notă artistică personală pentru a uni totul într-un ansamblu coerent.

Download

Github Proiect: https://github.com/antra-cet/Candy-Sorter

Link video youtube: https://youtu.be/xA97N-r_17E

Jurnal

10 aprilie - Stabilire tema proiect

20 aprilie - Achizitionare componente

5 Mai - Pagina documentatie OCW

17 Mai - Hardware

26 Mai - Software

29 Mai - Finalizare pagina documentatie

Bibliografie/Resurse

- Miscare Servo Motoare: https://youtu.be/QbgTl6VSA9Y
- Calibrare si folosire senzor culoare: https://youtu.be/MwdANEcTiPY
- Constructia casei: https://youtu.be/rBc1GZu7CeA
- Conectare LCD: https://howtomechatronics.com/tutorials/arduino/lcd-tutorial/

Export to PDF