Introduction to Arduino and Controlling a 7-Segment Display

Yihan Liao, Fleming society

November 28, 2023

Overview

Welcome to this introductory guide on Arduino programming, focusing on controlling a 7-segment LED display. This guide is designed for students with little to no experience in Arduino coding. By the end of this tutorial, you will understand how to write and use Arduino code to control a 7-segment display to show numbers and cycle through them.

The 7-Segment Display

A 7-segment display is a form of electronic display device for displaying decimal numerals. It's an easy way to display numbers using a single device. Each segment of the display can be controlled individually to display the numbers 0-9.

Required Components

1. Arduino Uno Board
2. 7-Segment Display (Common Cathode Type)
3. Jumper Wires
4. Breadboard
5. Resistors (if required for limiting current to the LEDs)

Setting Up Your Arduino Environment

- Open the Arduino IDE
- Connect the Arduino Board: Connect your Arduino Uno to your computer using a USB cable.

Understanding the Code

Below is the example code that you will be using. This code is designed to control a 7-segment display to show numbers from 0 to 9 and cycle through different positions if it's a multi-digit display.

Details that you need to know:

- **Pin Definitions:** These lines define the Arduino pins connected to each segment of the 7-segment display. Each segment is labeled from 'A' to 'G' and 'Dp' for the decimal point. Additionally, pins for each digit on the display are defined.
- **Timer Variables:** These variables are used for timing control in the loop() function.
- **setup() Function:** This function initializes each defined pin as an output. It's called once when the Arduino starts.
- **loop() Function:** This is the main function of the Arduino. It repeatedly executes and controls the behavior of the display. It updates the displayed number and cycles through positions.
- **display_num() Function**: This custom function controls which segments are lit on the display based on the number to be shown and which position (digit) is active.

Task 1: Pin Definitions

Objective: Define the Arduino pins connected to the segments and digits of the 7-segment display.

```
const int A_pin = 9; // Segment A
const int B_pin = 8; // Segment B
const int C_pin = 7; // Segment C
const int D_pin = 6; // Segment D
const int E_pin = 5; // Segment E
const int F_pin = 4; // Segment F
const int G_pin = 3; // Segment G
const int Dp_pin = 2; // Decimal point (Dp) segment

const int Dig_1_pin = 13; // First digit
const int Dig_2_pin = 12; // Second digit
const int Dig_3_pin = 11; // Third digit
const int Dig_4_pin = 10; // Fourth digit
```

Explanation:

Each segment of the 7-segment display is controlled by an individual pin on the Arduino. The display has 7 segments labeled A to G, and a decimal point (Dp). Additionally, if it's a multi-digit display, each digit is also controlled by a separate pin.

Connecting the 7-Segment Display to Arduino

Connect Segment Pins: Connect each segment pin of the 7-segment display to the corresponding pin on the Arduino as defined in the code.

Connect Digit Pins: If using a multi-digit display, connect each digit control pin to its corresponding Arduino pin.

Common Ground: Ensure the common ground of the display is connected to the Arduino's ground.

Task 2: Timer Initialization

Objective: Set up timing variables for controlling the display timing.

```
unsigned long startMillis = 0;
unsigned long currentMillis = 0;
unsigned long startMillis_1 = 0;


int position = 1;
int number;
int delay_time = 500;
```

Explanation:

“startMillis” and **“startMillis_1”** are used for timing control in the **“loop()”** function. The **“position”** variable tracks the active digit position, **“number”** is the current number to be displayed, and **“delay_time”** sets the delay between number changes.

Task 3: Setup Function

Objective: Initialize each pin connected to the 7-segment display as an output.

```
void setup() {
  // set pinMode:
  pinMode(A_pin, OUTPUT);
  pinMode(B_pin, OUTPUT);
  pinMode(C_pin, OUTPUT);
  pinMode(D_pin, OUTPUT);
  pinMode(E_pin, OUTPUT);
  pinMode(F_pin, OUTPUT);
  pinMode(G_pin, OUTPUT);
  pinMode(Dp_pin, OUTPUT);
  pinMode(Dig_1_pin, OUTPUT);
  pinMode(Dig_2_pin, OUTPUT);
  pinMode(Dig_3_pin, OUTPUT);
  pinMode(Dig_4_pin, OUTPUT);
}
```

Explanation:

The **“setup()”** function runs once when the Arduino starts. It sets each pin connected to the display as an output using **“pinMode()”**.

Task 4: Main Loop

Objective: Control the display behavior, including number changes and digit position cycling.

```
void loop() {
  // Load Current Time
  currentMillis = millis();

  // Switch number
  if (currentMillis - startMillis_1 >= delay_time) {
    if (number == 9) {
      number = 0;
    }
    else {
      number ++;
    }
    startMillis_1 = currentMillis;
  }

  // Switch position
  if (currentMillis - startMillis >= 1) {
    if (position == 4) {
      position = 1;
    } else {
      position ++;
    }
    display_num (position, number);
    startMillis = currentMillis;
```

```
  }
}
```

Explanation:

The **“loop()”** function continuously runs, updating the display. It uses **“millis()”** for non-blocking timing control. The display number cycles from 0 to 9, and the digit position cycles through 1 to 4 if it's a multi-digit display.

Task 5: Display Function

Objective: Define how each number is displayed on the 7-segment display.

Structure of code:

```
void display_num(int pos, int num) {
  // Control segments based on the number
  switch (num) {
    case 0:
      digitalWrite(A_pin, HIGH); // Display the number 0
      // ... (Control other segments)
      break;
    // ... (Cases for other numbers)
  }

  // Activate the correct digit
  switch (pos) {
    case 1:
      digitalWrite(Dig_1_pin, LOW); // Activate first digit
      // ... (Deactivate other digits)
      break;
```

```
    // ... (Cases for other positions)
  }
}
```

Explanation:

The display_num() function controls which segments of the 7-segment display are lit up to form numbers 0-9. It also activates the appropriate digit position on multi-digit displays.

Here is a example of case 0 for your task try that and make sure that you understand how it works then try on the other cases:

Control segments based on the number:

```
case 0:
        digitalWrite(A_pin, HIGH);
        digitalWrite(B_pin, HIGH);
        digitalWrite(C_pin, HIGH);
        digitalWrite(D_pin, HIGH);
        digitalWrite(E_pin, HIGH);
        digitalWrite(F_pin, HIGH);
        digitalWrite(G_pin, LOW);
        digitalWrite(Dp_pin, LOW);
        break;
```

Activate the correct digit:

```
switch (pos) {
    case 1:
    digitalWrite(Dig_1_pin, LOW);
    digitalWrite(Dig_2_pin, HIGH);
    digitalWrite(Dig_3_pin, HIGH);
    digitalWrite(Dig_4_pin, HIGH);
      break;
```

Uploading and Running the Code

Type the above code into the Arduino IDE.

Compile the Code: Click the verify button (check mark) in the IDE to compile the code.

Upload the Code: With your Arduino connected, click the upload button (right arrow) in the IDE to upload the code to the Arduino.