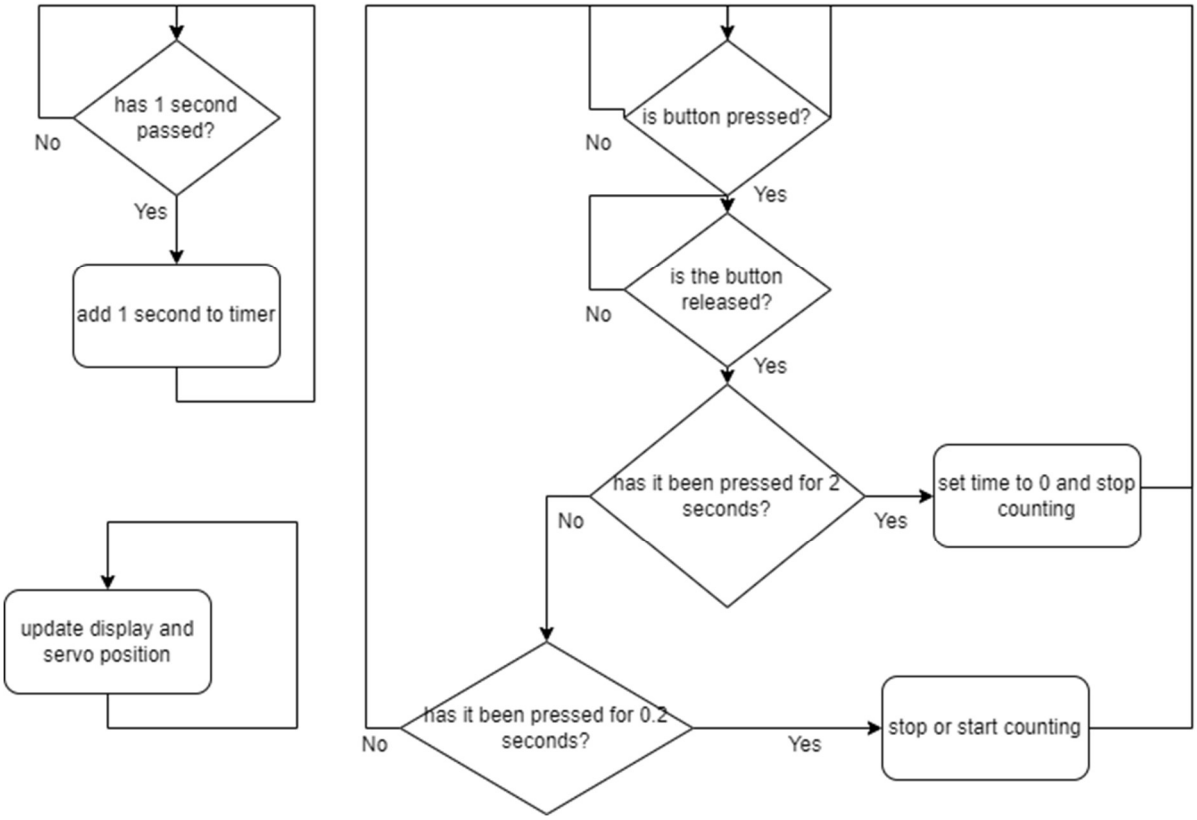
Arduino controlled dual display timer report

Project by: Zbigniew Barczyk

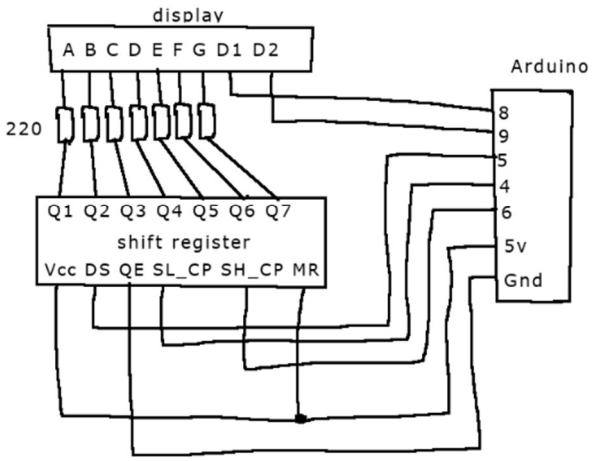
The goal is to design, create and program a circuit that will display a time up to 99 with a seven-segment display and 2 servos. The seven-segment display displays the number directly, while the servo is set to a specific position to indicate each digit of the number. To reduce the number of outputs used directly from the Arduino a bus expander chip and an I2C interface is used to control the display and servos. In addition, a button to reset the time as well as stop and start the count.

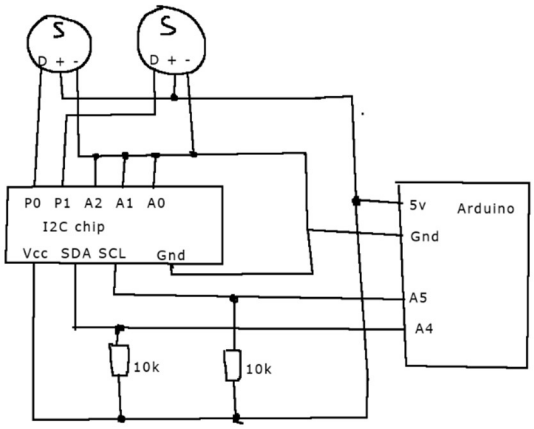
The four digit seven-segment display is a common cathode display, meaning each digit has a common ground connection. This can be used for multiplexing to display four different digits with only 11 wires going to the display – seven outputs for controlling the digit and four grounds for selectively choosing which digit to display at a given time. This can e further reduced to nine wires for two digits. To further reduce the required wires an 8-bit shift register is connected to the Arduino and the seven-segment display. This takes the wires going from the Arduino from the previous nine down to five if we power it indirectly. The shift registers take 3 wires from the Arduino to receive data and uses it to power the different segments of the display. To control different digits, each digit has its ground connected to a different Arduino pin, which is set to LOW to display the digit. Naturally each segment also has a 220-ohm resistor to increase the lifetime of the LED's.

The two servos are controlled using pulse-width modulation. Setting the servo pin to high for 1-2 microseconds every 20 microseconds sets the position of the servo between 0 and 180 degrees. To help with communicating with the servos an I2C expander chip is used to control them. The I2C chip takes two inputs from the Arduino and uses them to communicate to up to seven different devices per chip. This allows us to theoretically use less pins on the Arduino, but because we are only using two servos the numbers of pins used is the same. The I2C chip has its own library to make using it easier. It allows us to simply write on any of the seven pins on the chip without any issues. The servos are constantly connected to the power and ground.



This is the flow chart of the flow chart of the software running on the Arduino. It has 3 different loops. The smallest one constantly updates the display and the servo with the latest data. The second loop updates the data. It changes the time every second to simulate a clock. This allows the servos and display to display accurate data. The third loop is the button loop. It's responsible for the functionality of the button. It's triggered by an interrupt when the button is pressed. When it is released, it checks how long it was pressed and based on the duration takes an action. Pressed for longer than 200ms and shorter than 2s it stops the timer when its counting and starts the timer when its not counting, and when it's been pressed for longer than 2 seconds it resets the timer to 0 and stops it. Each of these loops happens independently of each other, but they interact with each other by altering variables.





These are the circuit diagrams for the display and servos. Both circuits are wired independently to the Arduino. They are using the same ground and 5v supply.

The 4-digit display has a 220-ohm resistor connected to each of the LEDs to prevent them from burning. The shift register is a SN74HC595 8-bit Shift register. It's connected to the Arduino with a ground, two 5v power wires and 4 other wires. DS is the data input, QE is the enable input pin, SH_CP is the shift register clock pin and SL_CP is the storage register clock pin. The Arduino sends an 8bit value to the register, which translates it into 7 pins being HIGH or LOW allowing us to control the 7-segment display. Pin 8 and 9 on the Arduino are set to low when we want to display that digit.

The servos are controlled with an PCF8574A 8-bit I/O I2C expander chip. The chip has 7 controllable pins (P). We are using P0 and P1 as outputs to give commands to the servos. The pins A0, A1 and A2 are the address pins. They are set to low here giving us the address 0x38, but they can be given any value for other addresses. The SDA and SCL are the data pins. They are connected to a 10k-ohm pullup resistor to reduce noise and eliminate incorrect data. The SCL pin is the clock pin, and the SDA pin is the data pin. In my program using the I2C library (PCF8574) I can select which pin (P0 or P1) I want to send data to. This allows us to easily communicate with the servo.

In addition to those two independent circuits there is a third one. It's a simple button connected to pin 2 on the Arduino and ground. The pin is configured as a pullup resistor. The button allows us to trigger interrupts in the code.

In my code I have two arrays containing values for the 7-segment display and servo. The display array contains a number which represents the number of the element on the display in the current wire configuration. The servo array contains delays for each number 2000 being '0' and 200 being '9'. I have a function for displaying digits on the display (up to four, currently two) and a function for positioning two servos. I also have a function for a button press that stops, resumes and resets the time countdown. It all comes together in the program to create a dual display.

CODE:

```
#include <PCF8574.h>

#define SDA A4   //analog  I2C chip
#define SCL A5   //analog I2C chip
#define Shift_Latch 4 //digital shift reg
#define Shift_Data 5    //digital shift reg
#define Shift_Clock 6 //digital shift reg
#define Dis1 8  //n1 led gnd
#define Dis2 9  //n2 led gnd
#define Dis3 10 //n3 led gnd    //for extra digit, unused
#define Dis4 11 //n4 led gnd    //for extra digit, unused
#define ButtonPin 2

const int segDisp[10] = {63, 6, 91, 79, 102, 109, 125, 7, 127, 111};
const int servDelay[10] = {2000, 1800, 1600, 1400, 1200, 1000, 800, 600, 400, 200};

PCF8574 PCF_01(0x38);
unsigned long nextMillis;
unsigned long currentMillis;
volatile unsigned long debounceTimer;
volatile unsigned long lastDebounceTimer = 0;
int timer;
boolean counting = false;



void setup() {

pinMode(SDA, OUTPUT);
pinMode(SCL, OUTPUT);
pinMode(Shift_Latch, OUTPUT);
pinMode(Shift_Data, OUTPUT);
pinMode(Shift_Clock, OUTPUT);
pinMode(Dis1, OUTPUT);
pinMode(Dis2, OUTPUT);
pinMode(Dis3, OUTPUT);
pinMode(Dis4, OUTPUT);
pinMode(ButtonPin, INPUT_PULLUP);
Serial.begin(9600);
PCF_01.begin();
int millis;
attachInterrupt(digitalPinToInterrupt(ButtonPin), buttonPress, CHANGE);

    digitalWrite(Dis1, HIGH);
    digitalWrite(Dis2, HIGH);
    digitalWrite(Dis3, HIGH);
    digitalWrite(Dis4, HIGH);
    timer = 0;

}

void loop()
{


 currentMillis = millis();

 if(counting == true)    //are we counting?
 {
   if(currentMillis >= nextMillis) //clock
   {
     nextMillis = currentMillis + 1000;
     if(timer<99)
       timer=timer+1;
     else
       timer=0;
   }
 }
```

```
 displayTime(timer);   //updates servo and displays 7 segment display value


}


void allDigit(int a, int b)   //void allDigit(int a, int b, int c, int d) //4 digit version //mothod for controlling display, for 2 also for 4
{
  digitalWrite(Shift_Latch, LOW);
   shiftOut(Shift_Data, Shift_Clock, MSBFIRST, segDisp[a]);
   digitalWrite(Shift_Latch, HIGH);
   digitalWrite(Dis1, LOW);
   digitalWrite(Dis1, HIGH);

   digitalWrite(Shift_Latch, LOW);
   shiftOut(Shift_Data, Shift_Clock, MSBFIRST, segDisp[b]);
   digitalWrite(Shift_Latch, HIGH);
   digitalWrite(Dis2, LOW);
   digitalWrite(Dis2, HIGH);

//commentd out because time goes to 99 only, kept in case that changes
//   digitalWrite(Shift_Latch, LOW);
//   shiftOut(Shift_Data, Shift_Clock, MSBFIRST, c);
//   digitalWrite(Shift_Latch, HIGH);
//   digitalWrite(Dis3, LOW);
//   digitalWrite(Dis3, HIGH);
//
//
//   digitalWrite(Shift_Latch, LOW);
//   shiftOut(Shift_Data, Shift_Clock, MSBFIRST, d);
//   digitalWrite(Shift_Latch, HIGH);
//   digitalWrite(Dis4, LOW);
//   digitalWrite(Dis4, HIGH);
}
void allMotors(int a, int b)    //method for controlling both servos
{
    PCF_01.write(0, HIGH);
    delayMicroseconds(servDelay[a]);
    PCF_01.write(0, LOW);
    delayMicroseconds(20000-servDelay[a]);

    PCF_01.write(1, HIGH);
    delayMicroseconds(servDelay[b]);
    PCF_01.write(1, LOW);
    delayMicroseconds(20000-servDelay[b]);
}

void buttonPress()  //interrupt method for button debounce
{
 debounceTimer = millis();
 if((debounceTimer - lastDebounceTimer)>= 2000 && digitalRead(ButtonPin) == 1)
 {
     timer=0;
     counting = false;
 }
 else if((debounceTimer - lastDebounceTimer)>= 200 && digitalRead(ButtonPin) == 1)
 {
     counting = !counting;
 }
 lastDebounceTimer = debounceTimer;
}

void displayTime(int timeToDisplay)   //time can be between 0 and 99 //method for combining servo and display
{
 allDigit(timeToDisplay/10, timeToDisplay%10);
 allMotors(timeToDisplay/10, timeToDisplay%10);
}
//end of code
```