// Demonstration of LED multiplexing with Arduino

// Author: Nick Gammon

// Date: 2 December 2013

// Put a suitable resistor in series with each segment LED (eg. 180 ohm)

const byte PATTERN\_COUNT = 16;

const byte SEGMENTS = 7;

const byte DIGITS = 4;

const byte columnPins [SEGMENTS] = { 2, 3, 4, 5, 6, 7, 8 }; // a, b, c, d, e, f, g

const byte digitPins [DIGITS] = { 9, 10, 11, 12 }; // DIG1, DIG2, DIG3, DIG4

#define COMMON\_ANODE false // make false for common cathode LEDs

#if COMMON\_ANODE

// For common ANODE:

const byte SEGMENT\_ON = LOW;

const byte SEGMENT\_OFF = HIGH;

const byte DIGIT\_ON = HIGH;

const byte DIGIT\_OFF = LOW;

#else

// For common CATHODE:

const byte SEGMENT\_ON = HIGH;

const byte SEGMENT\_OFF = LOW;

const byte DIGIT\_ON = LOW;

const byte DIGIT\_OFF = HIGH;

#endif

// extra segment patterns (you can add more)

const byte SHOW\_HYPHEN = 0x0A;

const byte SHOW\_E = 0x0B;

const byte SHOW\_H = 0x0C;

const byte SHOW\_L = 0x0D;

const byte SHOW\_P = 0x0E;

const byte SHOW\_BLANK = 0x0F;

const PROGMEM byte digitSegments [PATTERN\_COUNT] =

{

0b1111110, // 0

0b0110000, // 1

0b1101101, // 2

0b1111001, // 3

0b0110011, // 4

0b1011011, // 5

0b1011111, // 6

0b1110000, // 7

0b1111111, // 8

0b1111011, // 9

0b0000001, // 0x0A -> -

0b1001111, // 0x0B -> E

0b0110111, // 0x0C -> H

0b0001110, // 0x0D -> L

0b1100111, // 0x0E -> P

0b0000000, // 0x0F -> blank

};

volatile byte numberToShow [DIGITS] = { SHOW\_H, SHOW\_E, SHOW\_L, 0 }; // HELO

// timer Interrupt Service Routine (ISR) to update the LEDs

ISR (TIMER2\_COMPA\_vect)

{

static byte digit = 0;

byte thisDigit = numberToShow [digit];

// check for out of range, if so show a blank

if (thisDigit >= PATTERN\_COUNT)

thisDigit = SHOW\_BLANK;

// turn off old digit

for (byte i = 0; i < DIGITS; i++)

digitalWrite (digitPins[i], DIGIT\_OFF);

// set segments

for (byte j = 0; j < SEGMENTS; j++)

digitalWrite (columnPins [j], // which segment pin

(pgm\_read\_byte (digitSegments + thisDigit) // get bit pattern

& bit (SEGMENTS - j - 1)) // see if set or not

? SEGMENT\_ON : SEGMENT\_OFF); // set appropriately (HIGH or LOW)

// activate this digit

digitalWrite (digitPins [digit], DIGIT\_ON);

// wrap if necessary

if (++digit >= DIGITS)

digit = 0;

} // end of TIMER2\_COMPA\_vect

void setup()

{

for (byte i = 0; i < SEGMENTS; i++)

pinMode(columnPins[i], OUTPUT); // make all the segment pins outputs

for (byte i = 0; i < DIGITS; i++)

pinMode(digitPins[i], OUTPUT); // make all the digit pins outputs

// set up to draw the display repeatedly

// Stop timer 2

TCCR2A = 0;

TCCR2B = 0;

// Timer 2 - gives us a constant interrupt to refresh the LED display

TCCR2A = bit (WGM21) ; // CTC mode

OCR2A = 63; // count up to 64 (zero relative!!!!)

// Timer 2 - interrupt on match at about 2 kHz

TIMSK2 = bit (OCIE2A); // enable Timer2 Interrupt

// start Timer 2

TCCR2B = bit (CS20) | bit (CS22) ; // prescaler of 128

delay (1000); // give time to read "HELO" on the display

} // end of setup

void loop ()

{

unsigned long elapsedSeconds = millis () / 1000;

char buf [10];

sprintf (buf, "%04ld", elapsedSeconds);

for (byte i = 0; i < DIGITS; i++)

numberToShow [i] = buf [i] & 0x0F;

} // end of loop