Introduction to robotics
5th lab

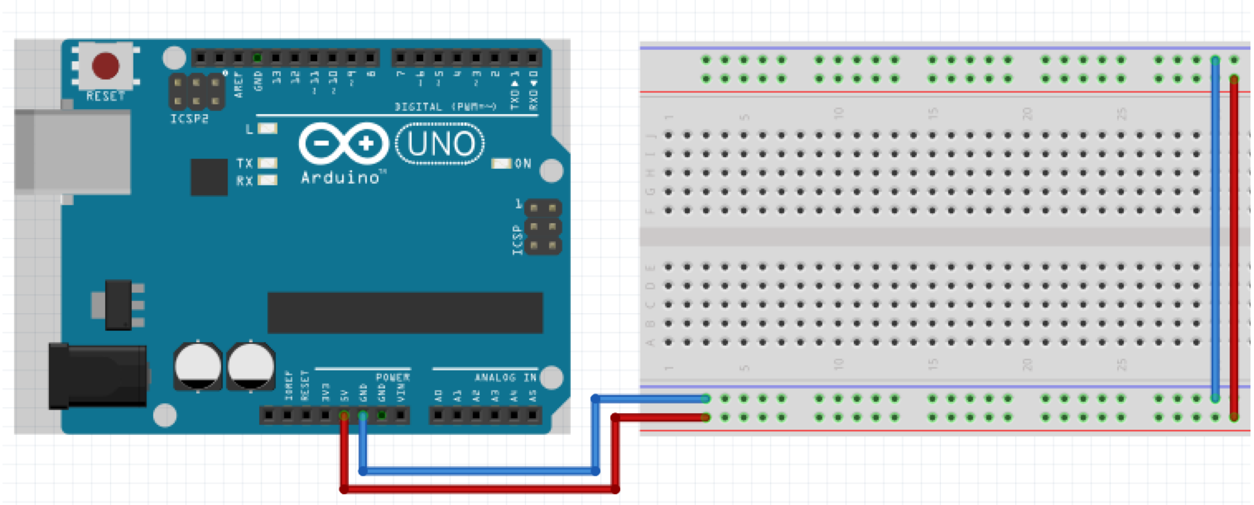
Remember, when possible, choose the wire color accordingly:

- **BLACK** (or dark colors) for **GND**
- **RED** (or colored) for **POWER (3.3V / 5V / VIN)**
- **Remember** than when you use digitalWrite or analogWrite, you actually send power over the PIN, so you can use the same color as for **POWER**
- **Bright Colored** for read signal
- We know it is not always possible to respect this due to lack of wires, but the first rule is **NOT USE BLACK FOR POWER OR RED FOR GND!**

Now, let’s pick it up where we left off…

Pull out your Arduino and breadboard and connect them like in the schematic. This is to “power up” the breadboard so we can easily have access to **5V** and **GND**.

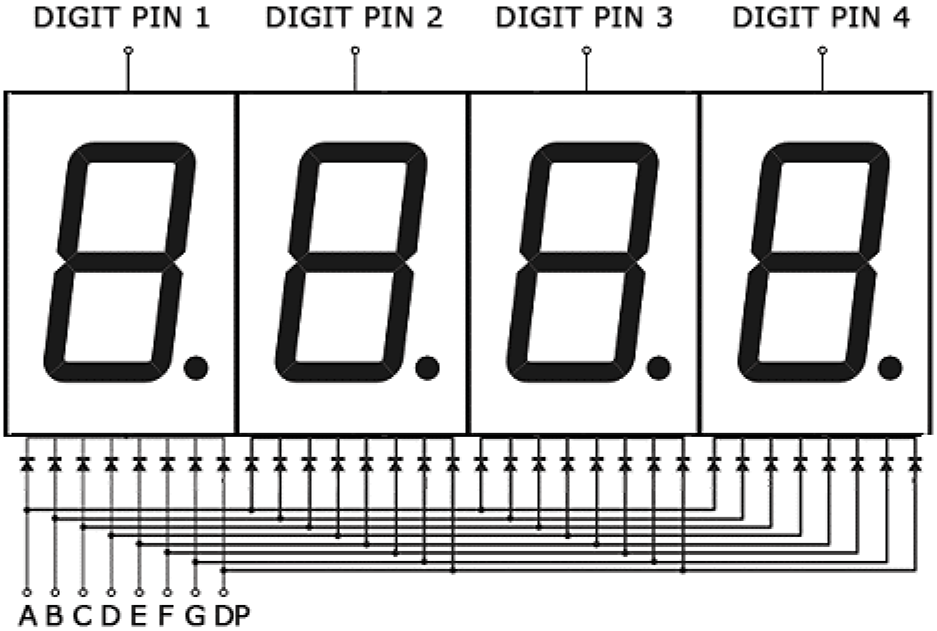
Attention! Remember how the breadboard works. Use correct wire colors.

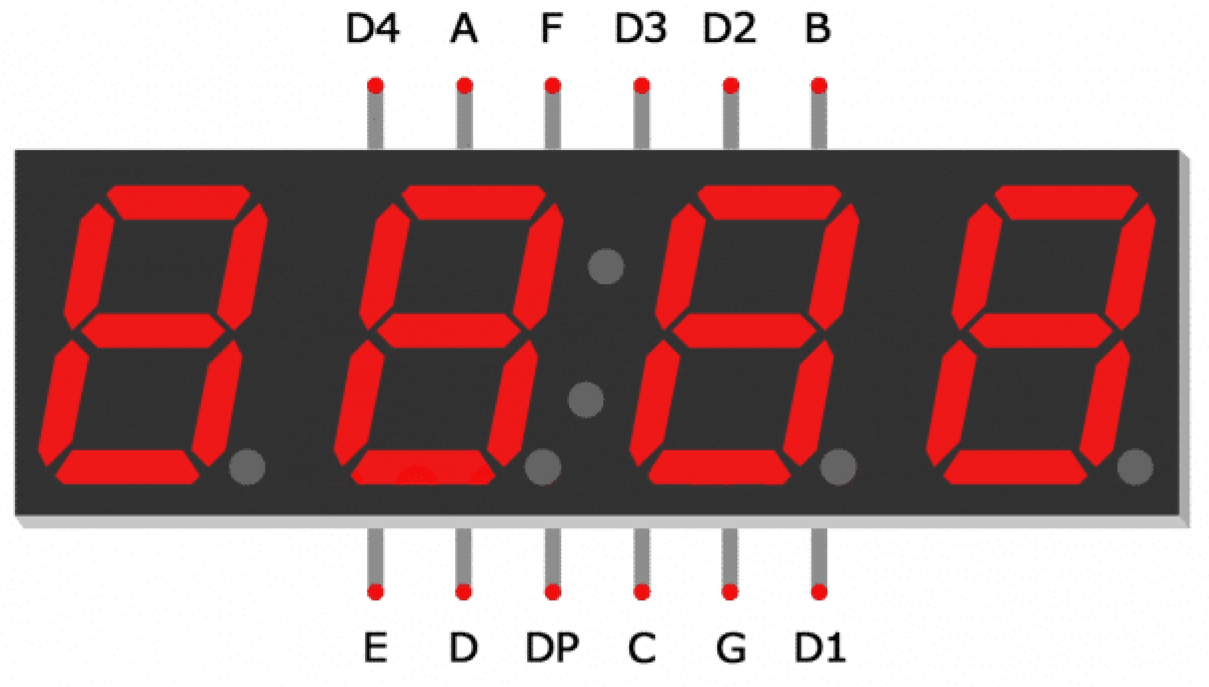


1. 4 digit 7-segment display

So far we have only worked with single digit 7-segment displays. To display information such as the time or temperature, you will want to use a 2 or 4 digit display, or connect multiple single digit displays side by side.

In multi-digit displays, one segment pin (A, B, C, D, E, F, G, and DP) controls the same segment on all of the digits. Multi-digit displays also have separate common pins for each digit. These are the digit pins. You can turn a digit on or off by switching digit pin.





Wiring diagram. D1 - D4 control the shown digits
Source: http://www.circuitbasics.com/arduino-7-segment-display-tutorial/

2. Shift Register: 74HC595

Consult the course and the datasheet for more details
https://www.diodes.com/assets/Datasheets/74HC595.pdf

- It's a sequential logic circuit
- Used for storage or transfer of binary data
- Can convert from serial to parallel data
- Used as memory or buffer stages within other chips
- For our use case, they allow us to use less pins in an Arduino

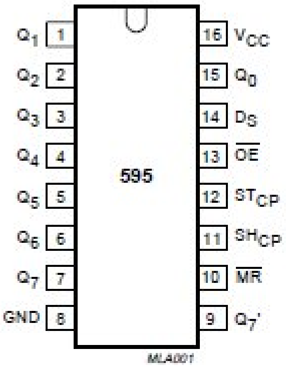


Fig.1 Pin configuration.

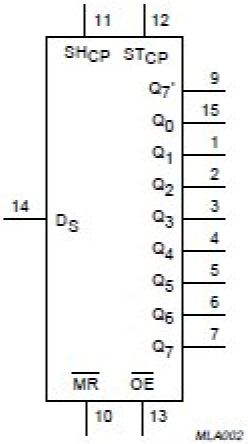
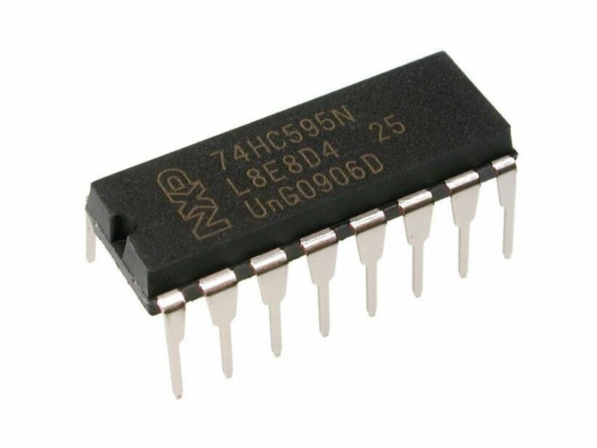
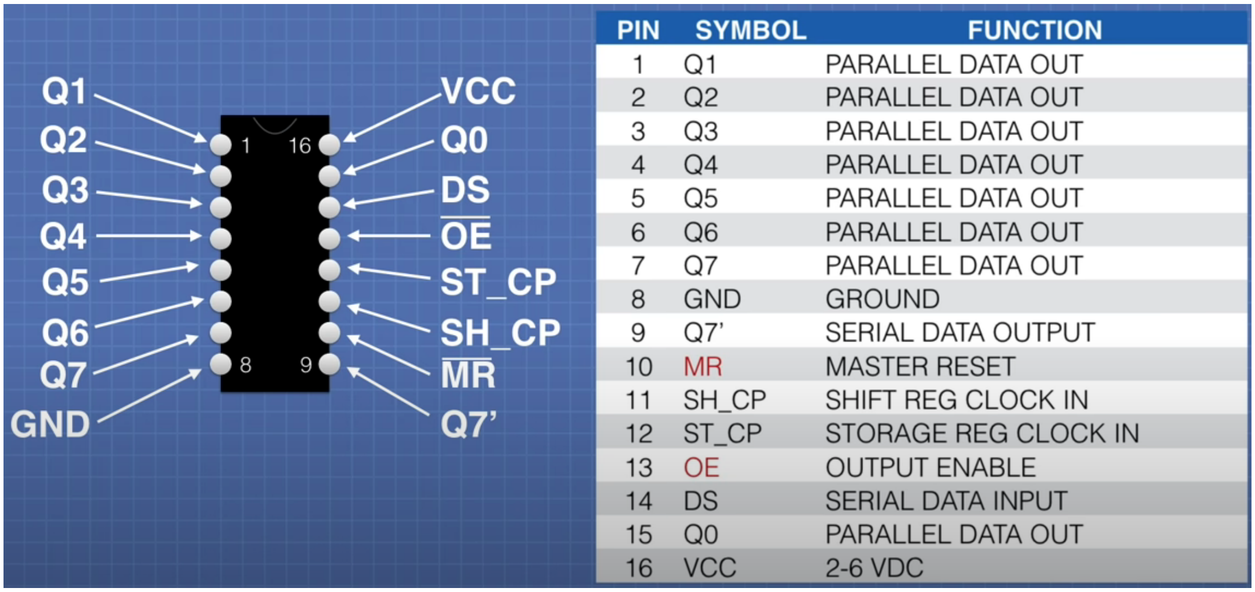


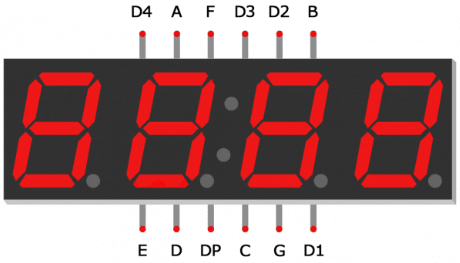
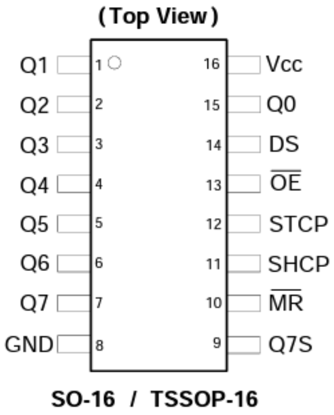
Fig.2 Logic symbol.

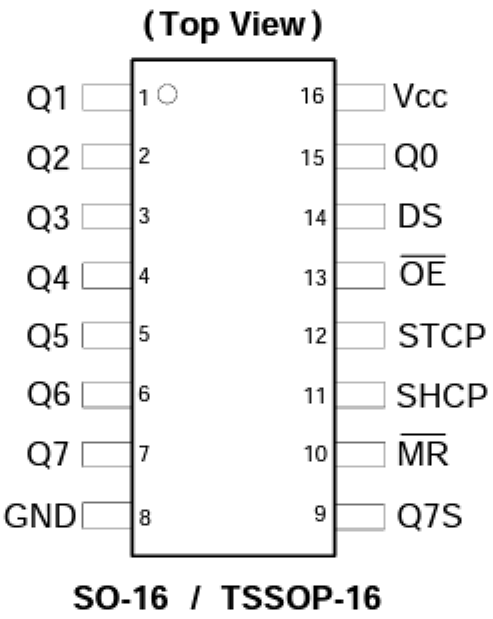




PIN	SYMBOL	FUNCTION
1	Q1	PARALLEL DATA OUT
2	Q2	PARALLEL DATA OUT
3	Q3	PARALLEL DATA OUT
4	Q4	PARALLEL DATA OUT
5	Q5	PARALLEL DATA OUT
6	Q6	PARALLEL DATA OUT
7	Q7	PARALLEL DATA OUT
8	GND	GROUND
9	Q7’	SERIAL DATA OUTPUT
10	MR	MASTER RESET
11	SH_CP	SHIFT REG CLOCK IN
12	ST_CP	STORAGE REG CLOCK IN
13	OE	OUTPUT ENABLE
14	DS	SERIAL DATA INPUT
15	Q0	PARALLEL DATA OUT
16	VCC	2-6 VDC

Connections table:

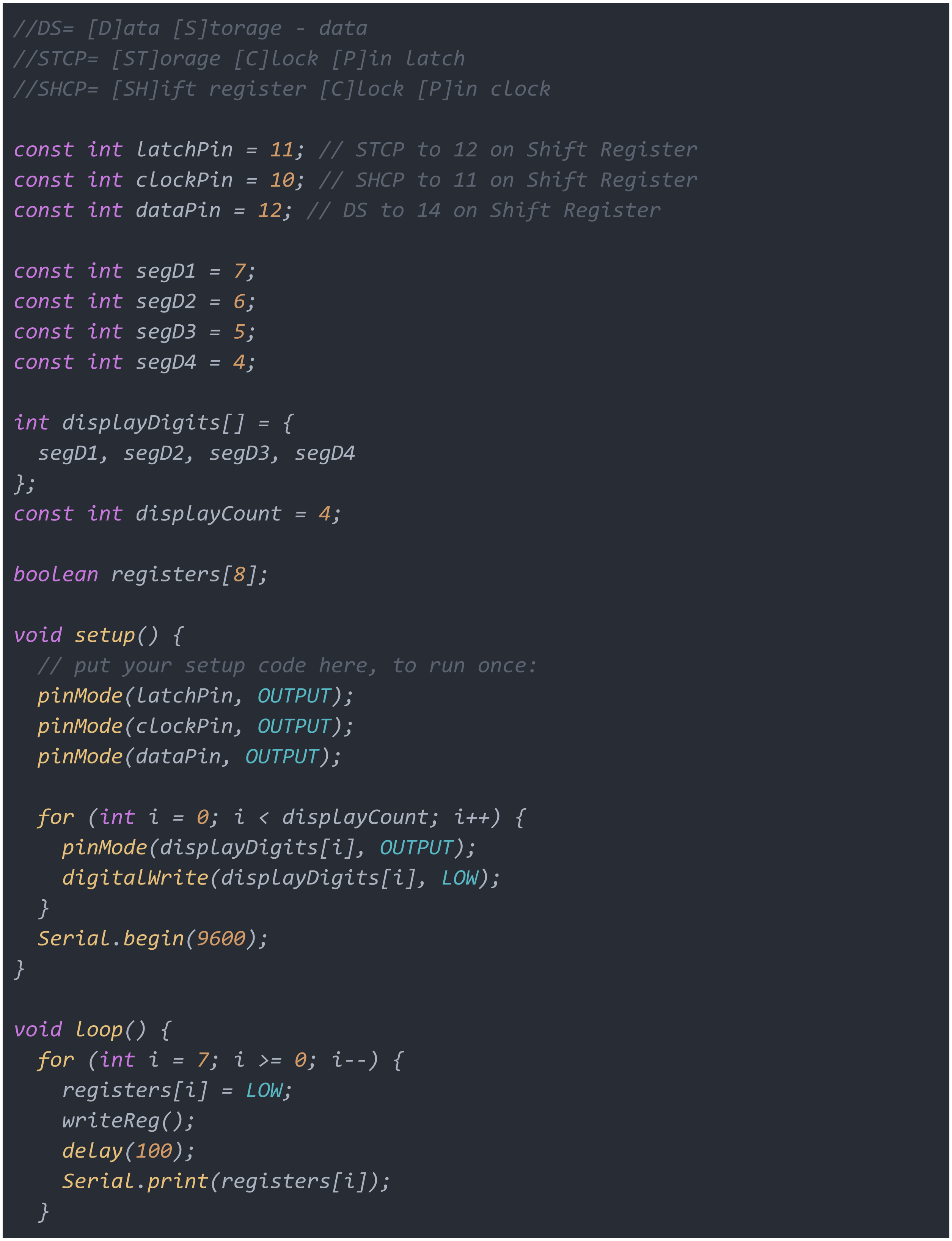
Shift Register PIN	Display PIN	Schematic	Schematic (1 digit)
Q7	A	 D4 A F D3 D2 B E D DP C G D1	 (Top View) Q1 Q2 Q3 Q4 Q5 Q6 Q7 GND Vcc Q0 DS OE STCP SHCP MR Q7S SO-16 / TSSOP-16
Q6	B		
Q5	C		
Q4	D		
Q3	E		
Q2	F		
Q1	G		
Q0	DP		

Shift Register PIN	Arduino PIN	Schematic (shift register)
DS	12	 (Top View) Q1 1 Q2 2 Q3 3 Q4 4 Q5 5 Q6 6 Q7 7 GND 8 16 Vcc 15 Q0 14 DS 13 $\overline{OE}$ 12 STCP 11 SHCP 10 $\overline{MR}$ 9 Q7S SO-16 / TSSOP-16
STCP	11	
SHCP	10	
GND	GND	
VCC	5V	
MR	5V	
OE	GND	

Display PIN	Arduino
D1	7
D2	6
D3	5
D4	4

Let’s code!

Example 1: cycle through all the segments, turning them all on and then all off. Writing manually to the shift register



```
//DS= [D]ata [S]torage - data
//STCP= [ST]orage [C]lock [P]in latch
//SHCP= [SH]ift register [C]lock [P]in clock

const int latchPin = 11; // STCP to 12 on Shift Register
const int clockPin = 10; // SHCP to 11 on Shift Register
const int dataPin = 12; // DS to 14 on Shift Register

const int segD1 = 7;
const int segD2 = 6;
const int segD3 = 5;
const int segD4 = 4;

int displayDigits[] = {
  segD1, segD2, segD3, segD4
};
const int displayCount = 4;

boolean registers[8];

void setup() {
  // put your setup code here, to run once:
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);

  for (int i = 0; i < displayCount; i++) {
    pinMode(displayDigits[i], OUTPUT);
    digitalWrite(displayDigits[i], LOW);
  }
  Serial.begin(9600);
}

void loop() {
  for (int i = 7; i >= 0; i--) {
    registers[i] = LOW;
    writeReg();
    delay(100);
    Serial.print(registers[i]);
  }
  Serial.println();

  for (int i = 0; i < 8; i++) {
    registers[i] = HIGH;
    writeReg();
    delay(100);
    Serial.print(registers[i]);
  }
  Serial.println();
}

void writeReg() {
  digitalWrite(latchPin, LOW);
  for (int i = 7; i >= 0; i--) {
    digitalWrite(clockPin, LOW);
    digitalWrite(dataPin, registers[i]);
    digitalWrite(clockPin, HIGH);
  }
  digitalWrite(latchPin, HIGH);
}
```

Example 2: upgrading our function to use ShiftOut, seeing multiplexing in action and writing a 4-digit number to the display.

```
//DS= [D]ata [S]torage - data
//STCP= [ST]orage [C]lock [P]in latch
//SHCP= [SH]ift register [C]lock [P]in clock
// Define Connections to 74HC595

const int latchPin = 11; // STCP to 12 on Shift Register
const int clockPin = 10; // SHCP to 11 on Shift Register
const int dataPin = 12; // DS to 14 on Shift Register

/*  See that the array is declared as int
 *  The B in front is the binary representation of the int number
 *  Instead of B11111100, which displays 0, we can write 252
 */
int digitArray[16] = {
//A B C D E F G DP
  B11111100, // 0
  B01100000, // 1
  B11011010, // 2
  B11110010, // 3
  B01100110, // 4
  B10110110, // 5
  B10111110, // 6
  B11100000, // 7
  B11111110, // 8
  B11110110, // 9
  B11101110, // A
  B00111110, // b
  B10011100, // C
  B01111010, // d
  B10011110, // E
  B10001110  // F
};
const int segD1 = 7;
const int segD2 = 6;
const int segD3 = 5;
const int segD4 = 4;
int displayDigits[] = {
  segD1, segD2, segD3, segD4
};
const int displayCount = 4;

void setup () {
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  for (int i = 0; i < displayCount; i++) {
    pinMode(displayDigits[i], OUTPUT);
    digitalWrite(displayDigits[i], LOW);
  }
  Serial.begin(9600);
}

unsigned long lastIncrement = 0;
unsigned long delayCount = 50;
unsigned long number = 0;

void loop() {
  // writeNumber(1234);
  writeNumber(number);

  if (millis() - lastIncrement > delayCount) {
    number++;
    number %= 10000;
    lastIncrement = millis();
  }
}

void writeReg(int digit) {
    // ST_CP LOW to keep LEDs from changing while reading serial data
    digitalWrite(latchPin, LOW);
    // Shift out the bits
    shiftOut(dataPin, clockPin, MSBFIRST, digit);
    /* ST_CP on Rising to move the data from shift register
     * to storage register, making the bits available for output.
     */
    digitalWrite(latchPin, HIGH);
}


void showDigit(int displayNumber) {
  // first, disable all the display digits
  for (int i = 0; i < displayCount; i++) {
    digitalWrite(displayDigits[i], HIGH);
  }
  // then enable only the digit you want to use now
  digitalWrite(displayDigits[displayNumber], LOW);
}


void writeNumber(int number) {
  int currentNumber = number;
  int displayDigit = 0;
  int lastDigit;

  while (currentNumber != 0) {
    // get the last digit of the number
    lastDigit = currentNumber % 10;
    // enable only the display digit for that
    showDigit(displayDigit);
    // send the number to the display
    writeReg(digitArray[lastDigit]);
    // increase the delay to see multiplexing in action
    delay(5);
    // increment the display digit
    displayDigit++;
    // eliminate the last digit of the number
    currentNumber /= 10;
  }
}
```