

Introduction to robotics

5th lab

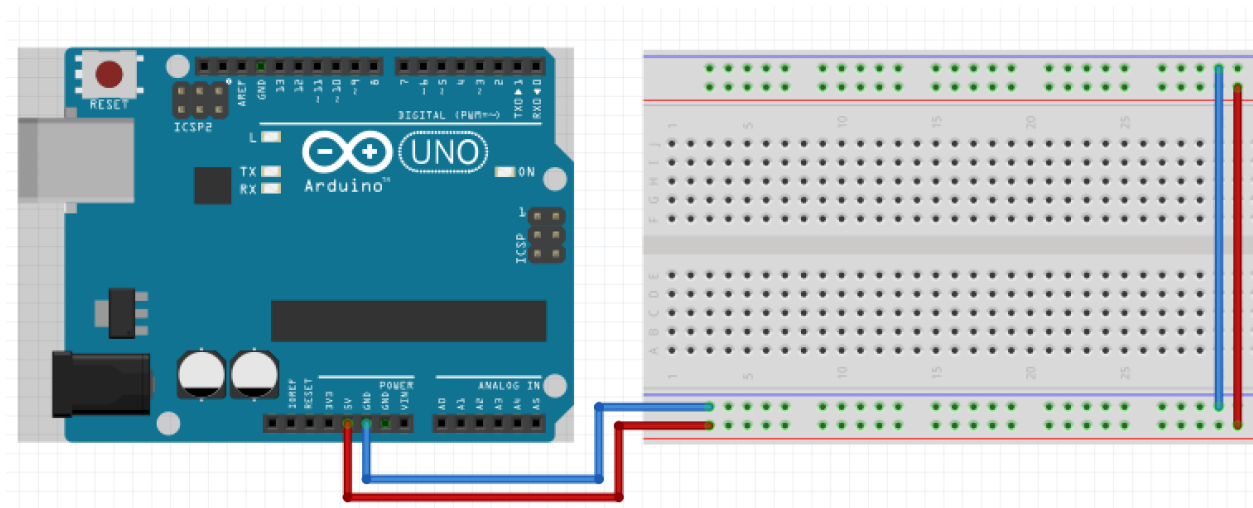
Remember, when possible, choose the wire color accordingly:

- **BLACK** (or dark colors) for **GND**
- **RED** (or colored) for **POWER (3.3V / 5V / VIN)**
- **Remember** that 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.

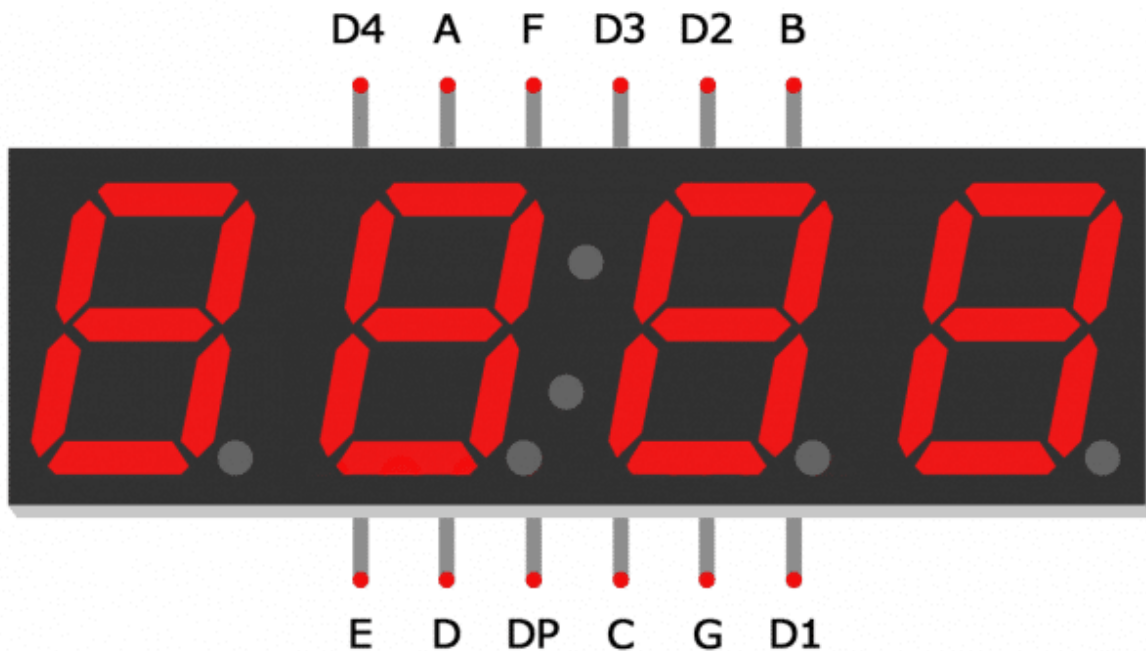


1. 4 digit 7-segment display

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

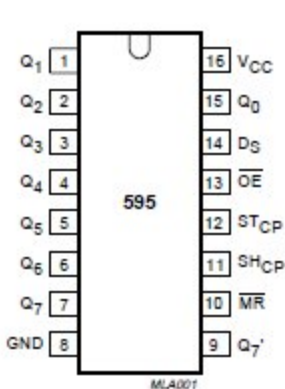
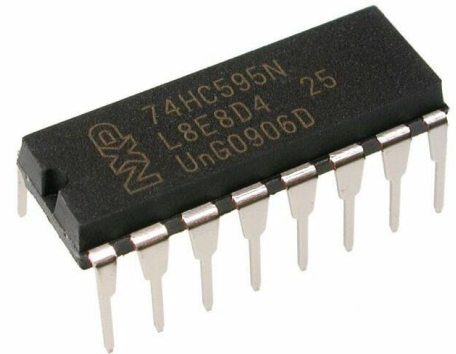


Fig.1 Pin configuration.

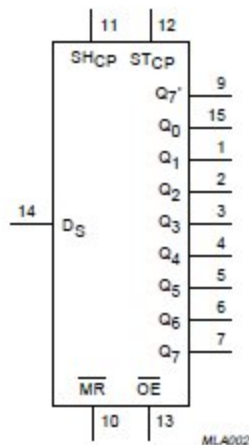
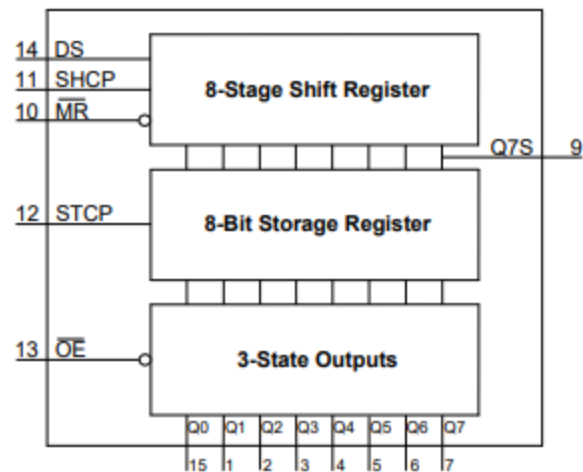
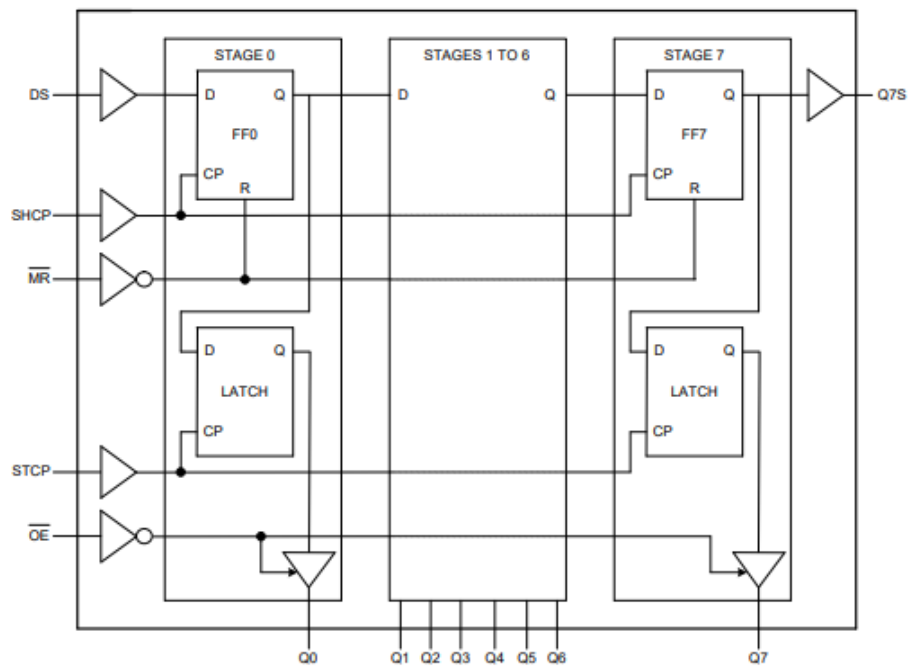


Fig.2 Logic symbol.

Functional Diagram

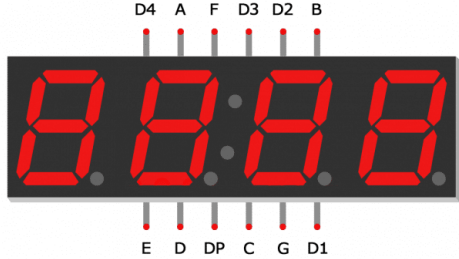
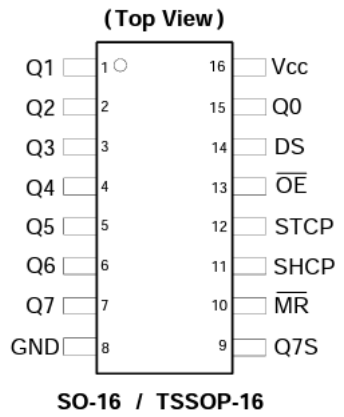


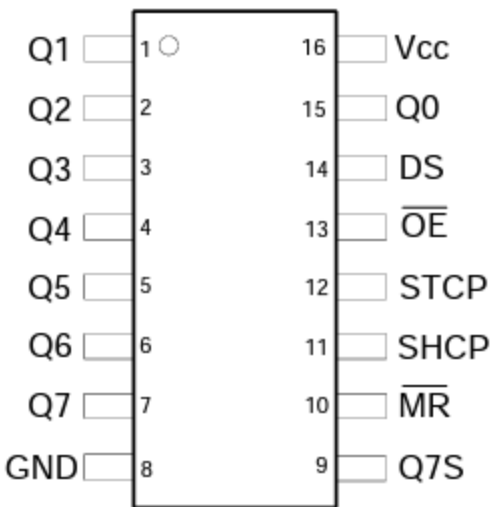
Logic Diagram



		PIN	SYMBOL	FUNCTION
Q1		1	Q1	PARALLEL DATA OUT
Q2		2	Q2	PARALLEL DATA OUT
Q3		3	Q3	PARALLEL DATA OUT
Q4		4	Q4	PARALLEL DATA OUT
Q5		5	Q5	PARALLEL DATA OUT
Q6		6	Q6	PARALLEL DATA OUT
Q7		7	Q7	PARALLEL DATA OUT
GND		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		<p>(Top View)</p>  <p>SO-16 / TSSOP-16</p>
Q6	B		
Q5	C		
Q4	D		
Q3	E		
Q2	F		
Q1	G		
Q0	DP		

Shift Register PIN	Arduino PIN	Schematic (shift register)
DS	12	<p>(Top View)</p>  <p>SO-16 / TSSOP-16</p>
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!

2.1 Cycle through all the segments, turning them all on and off. Write bits manually.

```
//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;

const byte regSize = 8; // 1 byte aka 8 bits

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

byte registers[regSize];
```

```
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 = regSize - 1; i >= 0; i--) {
        registers[i] = LOW;
        writeReg(registers);
        delay(100);
        Serial.print(registers[i]);
    }
    Serial.println();

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

void writeReg(byte encoding[]) {
    digitalWrite(latchPin, LOW);
    for (int i = 0; i < regSize; i++) { // MSBFIRST
        digitalWrite(clockPin, LOW);
        digitalWrite(dataPin, encoding[i]);
        digitalWrite(clockPin, HIGH);
    }
    digitalWrite(latchPin, HIGH);
}
```

2.2 Use an encoding matrix to pass symbols to the shift register. Write bits manually.

<missing code>

2.3 Use byte encodings and bit referencing to pass encodings to the shift register. Write bits manually.

<missing code>

2.3 Use byte encodings and bit referencing to pass encodings to the shift register. Write bits using shiftOut function. 2.4 Create a function that uses multiplexing to write on each display digit. Display a timer using millis().

<missing code>

3. Resources

- Persistence of vision: https://en.wikipedia.org/wiki/Persistence_of_vision
- Multiplexing: <https://en.wikipedia.org/wiki/Multiplexing>
- Shift register on arduino.cc (with daisy chain):
<https://www.arduino.cc/en/Tutorial/Foundations/ShiftOut>
- Shiftout function:
<https://www.arduino.cc/reference/en/language/functions/advanced-io/shiftout/>
- Good explanation of shift register on youtube (highly recommended):
<https://www.youtube.com/watch?v=Ys2fu4NINrA>
- Datasheet on 74HC595: <https://www.diodes.com/assets/Datasheets/74HC595.pdf>