Introduction to robotics
4th lab

First of all, **homework check**. Today we check for:

1. Github structure
2. Homework no. 4 (buzzer)

ToDo:

1. Provide link for github (for TA: insert it into the centralized document)
2. Leave the 8x8 matrix if you haven't already
3. Leave the 3rd potentiometers as well

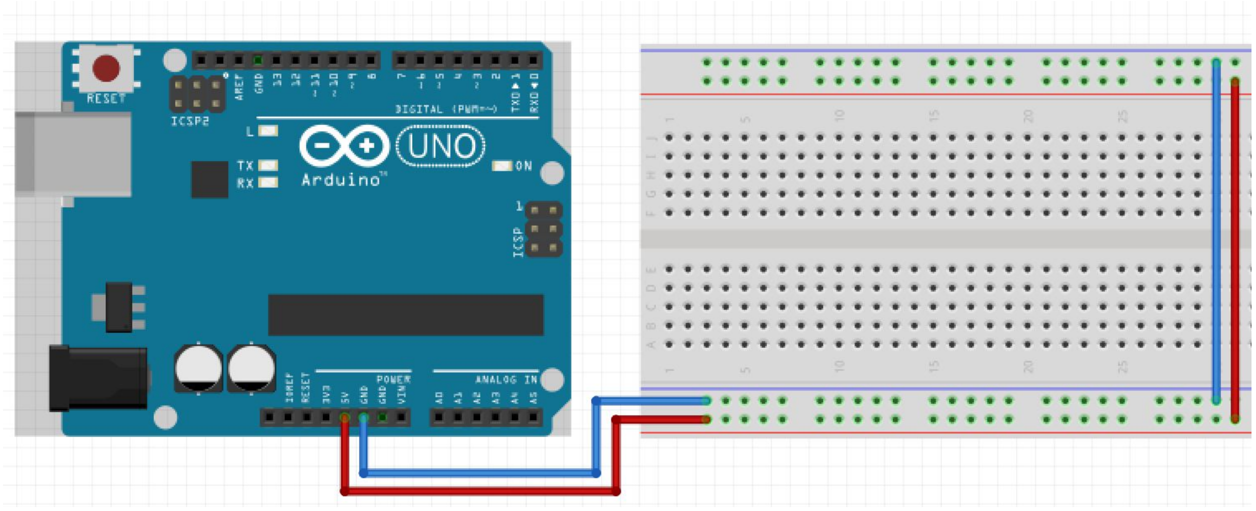
Remember, when possible, choose the wire color accordingly:

- **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 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.



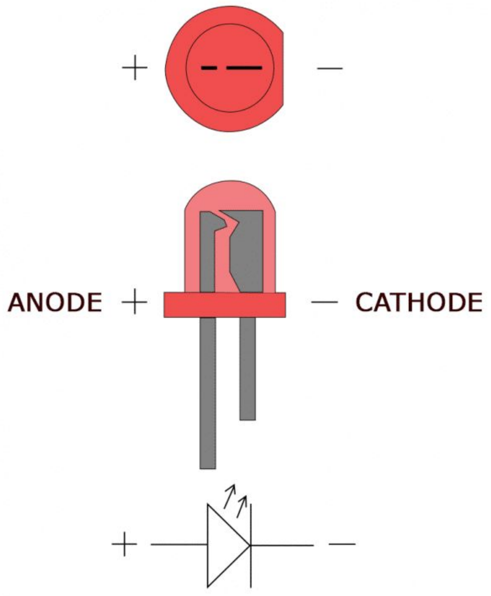
1. 7-segment display

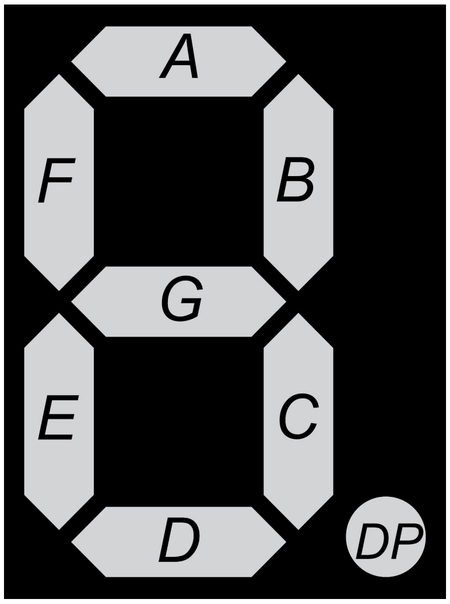
1.1 Recap

Questions:

1. **What is actually a RGB LED?**
    a. **A:** a combination of 3 LEDs in just one package.
2. **So, what's a 7-segment display?**
    a. **A:** short version, a combination of 8 LEDs in just one package.

Remember the LED basics:



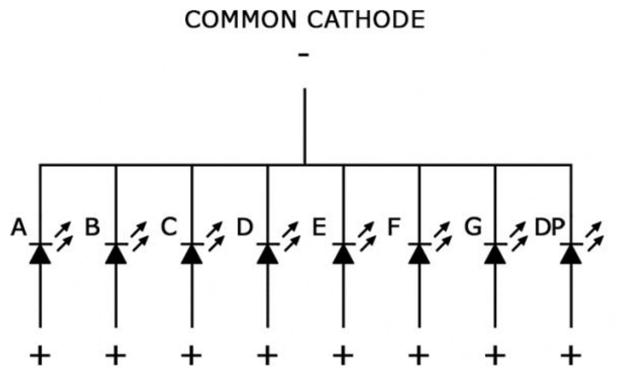


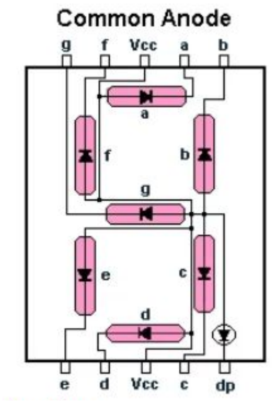
(source: https://en.wikipedia.org/wiki/Seven-segment_display)

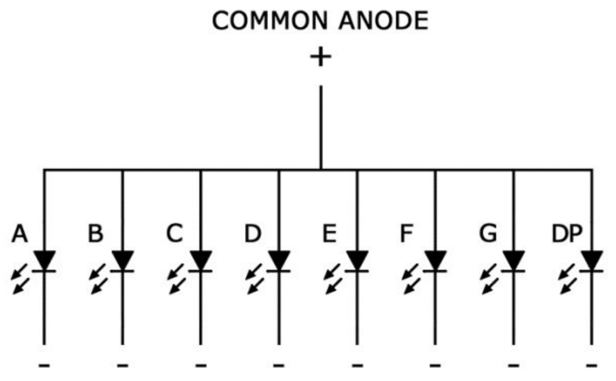
1.2 Introduction

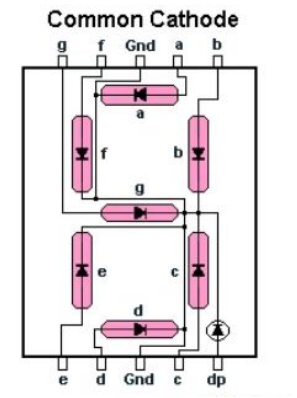
The 7-segment display, also written as “seven segment display”, consists of seven LEDs (hence its name) arranged in a rectangular fashion as shown. Each of the seven LEDs is called a segment because when illuminated the segment forms part of a numerical digit (both Decimal and Hex) to be displayed. An additional 8th LED is sometimes used within the same package thus allowing the indication of a decimal point (DP) when two or more 7-segment displays are connected together to display numbers greater than ten.

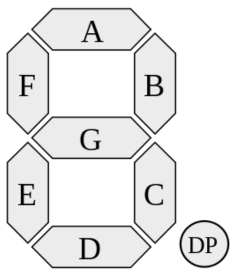
Like the RGB LED, 7-segment displays can have a **common cathode** or **common anode**. Although you most likely have common cathode, it’s best to check it with a multimeter.





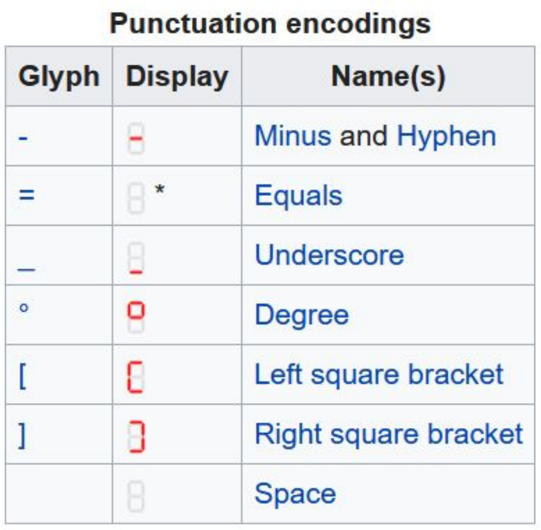






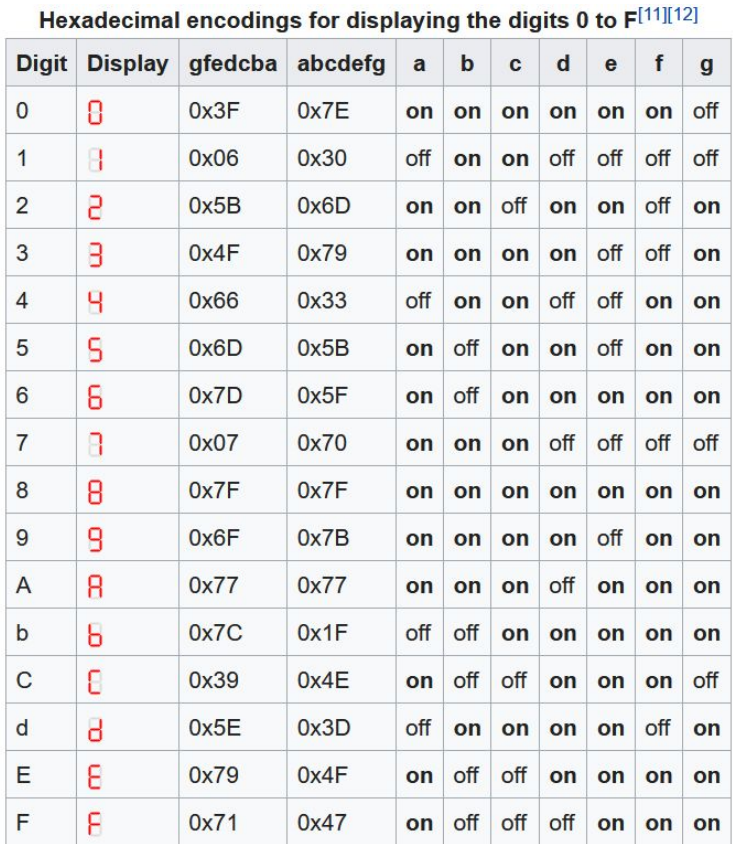
1.3 Displaying characters

The display system is mainly used for numbers and not letters. That is why some letters are not compatible and are easily mistaken for a number. Short messages giving status information (e.g. "no dISC" on a CD player) are also commonly represented on 7-segment displays. In the case of such messages it is not necessary for every letter to be unambiguous, merely for the words as a whole to be readable



Punctuation encodings

Glyph	Display	Name(s)
-		Minus and Hyphen
=	*	Equals
_		Underscore
°		Degree
[		Left square bracket
]		Right square bracket
		Space

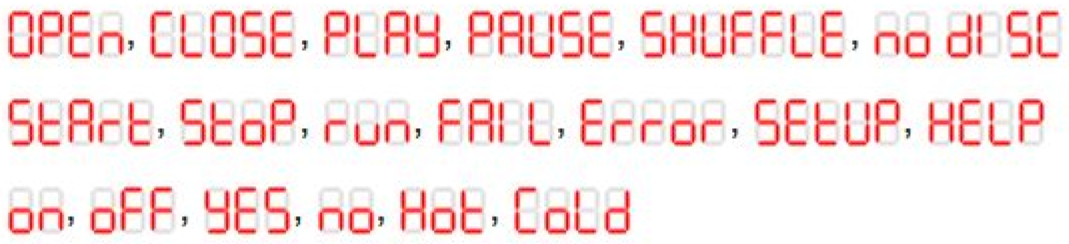


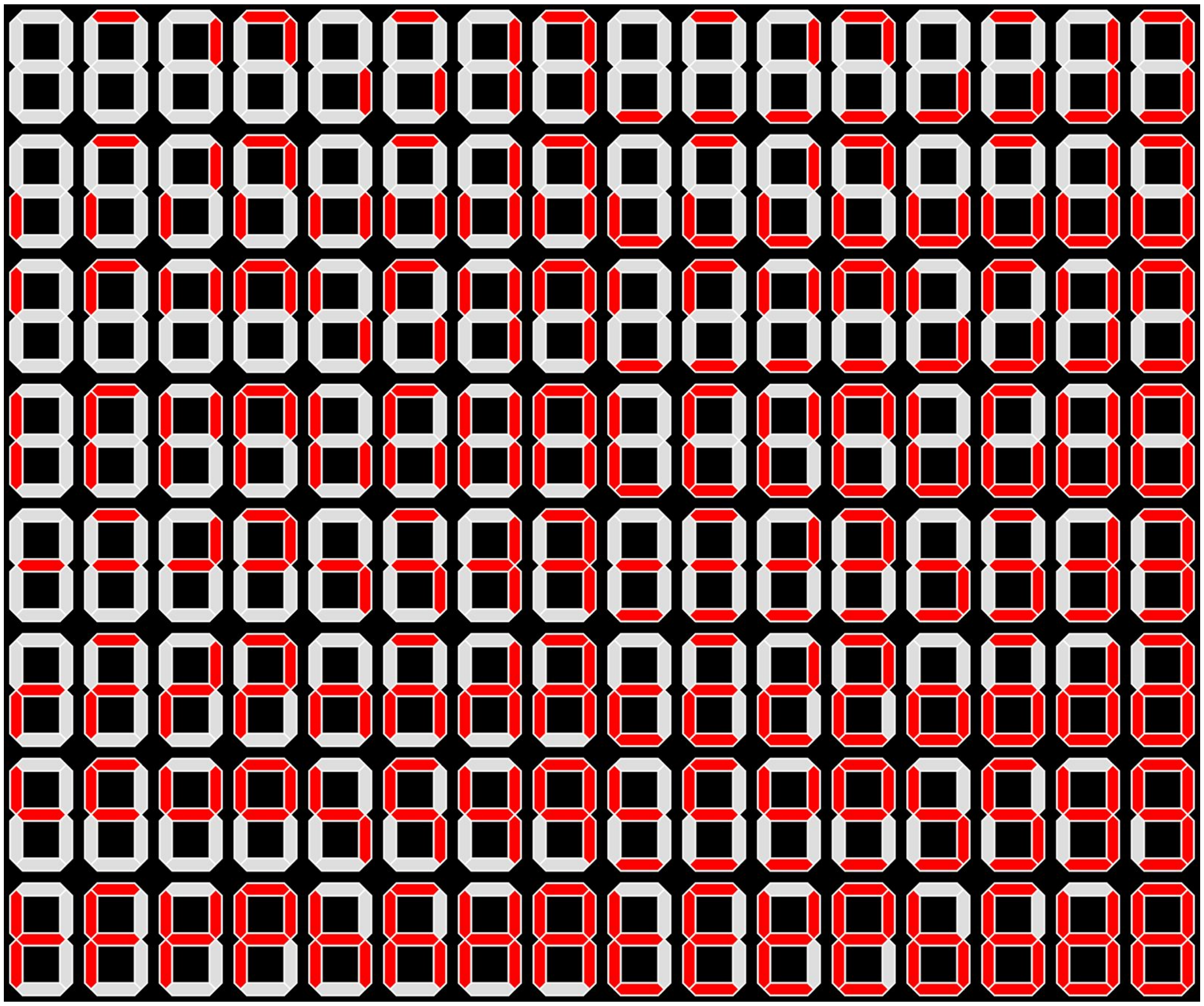
Hexadecimal encodings for displaying the digits 0 to F[11][12]

Digit	Display	gfedcba	abcdefg	a	b	c	d	e	f	g
0		0x3F	0x7E	on	on	on	on	on	on	off
1		0x06	0x30	off	on	on	off	off	off	off
2		0x5B	0x6D	on	on	off	on	on	off	on
3		0x4F	0x79	on	on	on	on	off	off	on
4		0x66	0x33	off	on	on	off	off	on	on
5		0x6D	0x5B	on	off	on	on	off	on	on
6		0x7D	0x5F	on	off	on	on	on	on	on
7		0x07	0x70	on	on	on	off	off	off	off
8		0x7F	0x7F	on	on	on	on	on	on	on
9		0x6F	0x7B	on	on	on	on	off	on	on
A		0x77	0x77	on	on	on	off	on	on	on
b		0x7C	0x1F	off	off	on	on	on	on	on
C		0x39	0x4E	on	off	off	on	on	on	off
d		0x5E	0x3D	off	on	on	on	on	off	on
E		0x79	0x4F	on	off	off	on	on	on	on
F		0x71	0x47	on	off	off	off	on	on	on

(source: https://en.wikipedia.org/wiki/Seven-segment_display).

Example of short messages:


OPEn, CLOSE, PLAY, PAUSE, SHUFFLE, no dISC
StArt, StoP, run, FAIL, Error, SEtUP, HELP
on, oFF, YES, no, Hot, CoLd



16x8-grid showing the 128 states of a seven-segment display

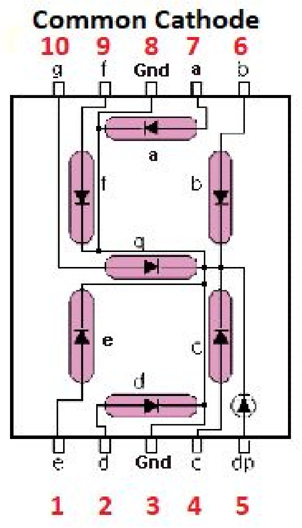
To determine which type of display you have, probe the pins with a multimeter. Try connecting the middle pin to the **COM - black -** of the multimeter, and any other pin (except the middle one on the other side) to the **V+ - red - .** If any of the segments stat lighting, it's a common cathode. If not, try reversing the wires; if any of the segments start lighting then it's a common anode.

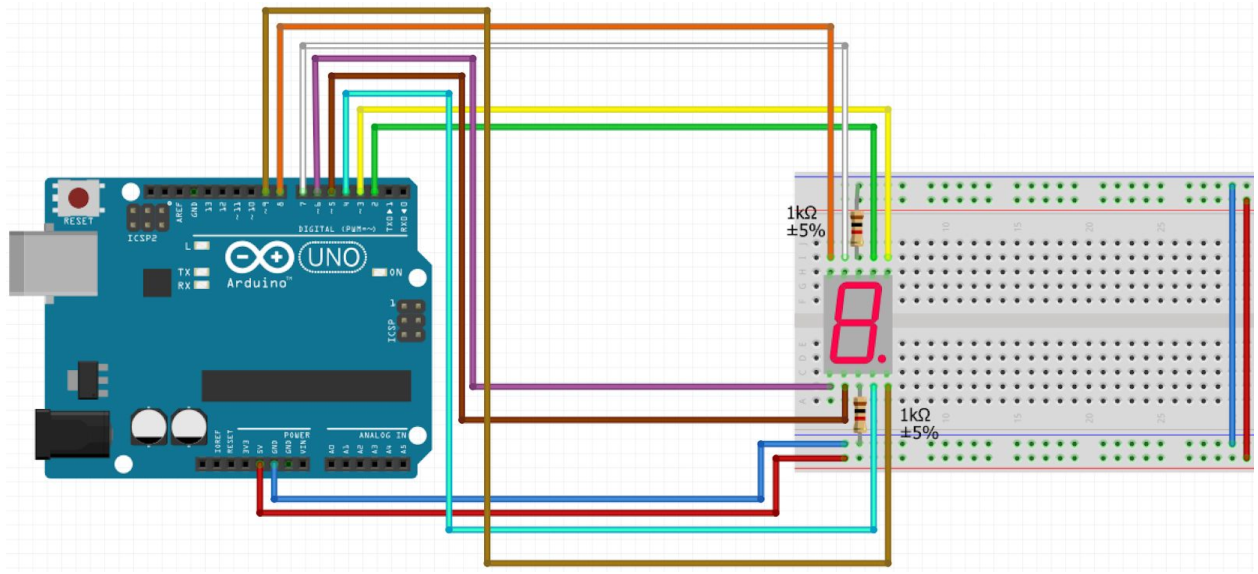
Questions:

1. What type of display do you have?

1.4 Connecting the 7-segment display

First of all, check which type of display you have. Common cathode must have Ax at the end. (x can have any value). If you don't have common cathode, ask for another one.
Common cathode: (5161Ax)
Connecting the 7 segment display to the Arduino board:
Attention! The pins have a corresponding letter and a number based on their circular order, starting with 1 from the bottom left. The red numbers are **NOT** the corresponding Arduino pins.

Display PIN	Arduino PIN	Schematic
A (7)	2	1  Common Cathode 10 9 8 7 6 g f Gnd a b a f b g e c d e d Gnd c dp 1 2 3 4 5
B (6)	3	
C (4)	4	
D (2)	5	
E (1)	6	
F (9)	7	
G (10)	8	
DP (5)	9	



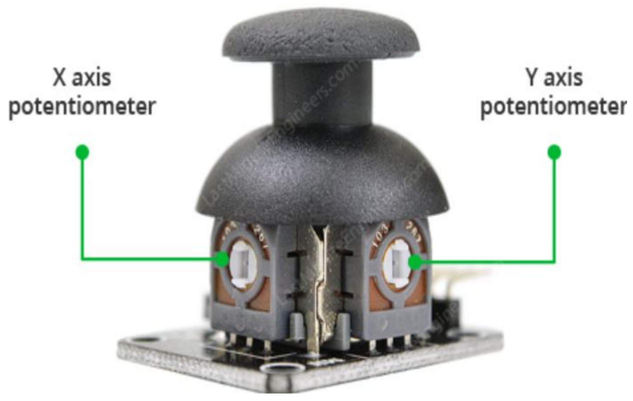
```
// declare all the segments pins
const int pinA = 2;
const int pinB = 3;
const int pinC = 4;
const int pinD = 5;
const int pinE = 6;
const int pinF = 7;
const int pinG = 8;
const int pinDP = 9;

const int segSize = 8;

int index = 0;
int state = HIGH;

// store the pins in an array for easier access
int segments[segSize] = {
  pinA, pinB, pinC, pinD, pinE, pinF, pinG, pinDP
};

void setup() {
  // initialize all the pins
  for (int i = 0; i < segSize; i++) {
      pinMode(segments[i], OUTPUT);
  }
}
```

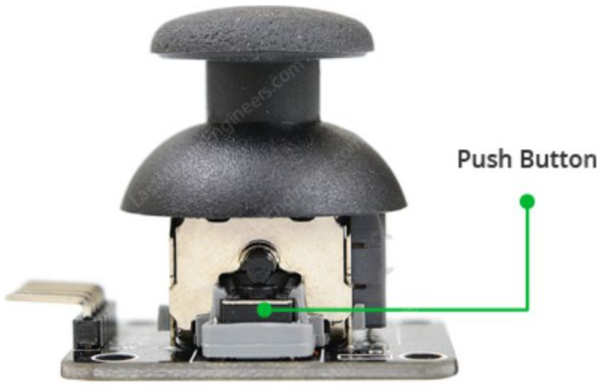
```
void loop() {
  // turn all the pins on in order. It is a good exercise to see if you
connected the wires properly
  digitalWrite(segments[index], state);
  index++;
  delay(500);
  if (index >= segSize) {
      index = 0;
      state = !state;
  }
}
```

Count from 0 to 9 (will need some explaining, don't rush it)

```
// declare all the segments pins
const int pinA = 2;
const int pinB = 3;
const int pinC = 4;
const int pinD = 5;
const int pinE = 6;
const int pinF = 7;
const int pinG = 8;
const int pinDP = 9;

const int segSize = 8;
const int noOfDigits = 10;

int dpState = LOW;
int index = 0;


// store the pins in an array for easier access
int segments[segSize] = {
  pinA, pinB, pinC, pinD, pinE, pinF, pinG, pinDP
};

byte digitMatrix[noOfDigits][segSize - 1] = {
// a  b  c  d  e  f  g
  {1, 1, 1, 1, 1, 1, 0}, // 0
  {0, 1, 1, 0, 0, 0, 0}, // 1
  {1, 1, 0, 1, 1, 0, 1}, // 2
  {1, 1, 1, 1, 0, 0, 1}, // 3
  {0, 1, 1, 0, 0, 1, 1}, // 4
  {1, 0, 1, 1, 0, 1, 1}, // 5
  {1, 0, 1, 1, 1, 1, 1}, // 6
  {1, 1, 1, 0, 0, 0, 0}, // 7
  {1, 1, 1, 1, 1, 1, 1}, // 8
  {1, 1, 1, 1, 0, 1, 1}  // 9
};



void displayNumber(byte digit, byte decimalPoint) {
  for (int i = 0; i < segSize - 1; i++) {
      digitalWrite(segments[i], digitMatrix[digit][i]);
  }

  // write the decimalPoint to DP pin
  digitalWrite(segments[segSize - 1], decimalPoint);
}

void setup() {
  // initialize all the pins
  for (int i = 0; i < segSize; i++) {
      pinMode(segments[i], OUTPUT);
  }
}

void loop() {
  // turn all the pins on in order. It is a good exercise to see if you
connected the wires properly
  displayNumber(index, dpState);
  index++;
  delay(500);
  if (index == noOfDigits) {
      index = 0;
      dpState = !dpState;
  }
}
```

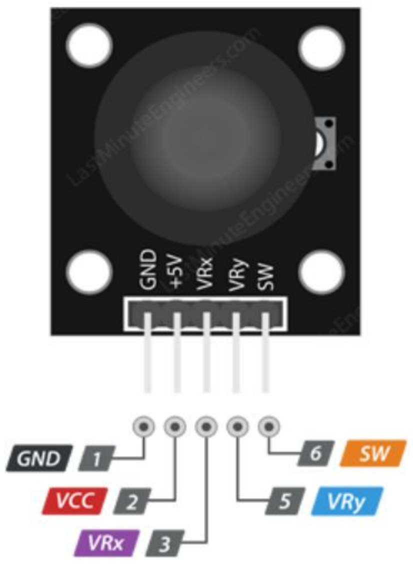
2. Joystick





The joystick is similar to two potentiometers connected together, one for the vertical movement (Y-axis) and other for the horizontal movement (X-axis). It also contains a switch which activates when you push down on the cap.
What we're gonna use in this lab is a self-centering spring loaded joystick, meaning when you release the joystick it will center itself. It also contains a comfortable cup-type knob/cap which gives the feel of a thumb-stick.

2.1 Pinout



GND is the Ground Pin
VCC supplies power for the module
VRx gives readout of the joystick in the horizontal direction (X-coordinate) i.e. how far left or right the joystick is pushed.
VRy gives readout of the joystick in the vertical direction (Y-coordinate) i.e. how far up and down the joystick is pushed.
SW is the output from the pushbutton. It's normally open, meaning the digital readout from the SW pin will be HIGH. When the button is pushed, it will connect to GND, giving output LOW.

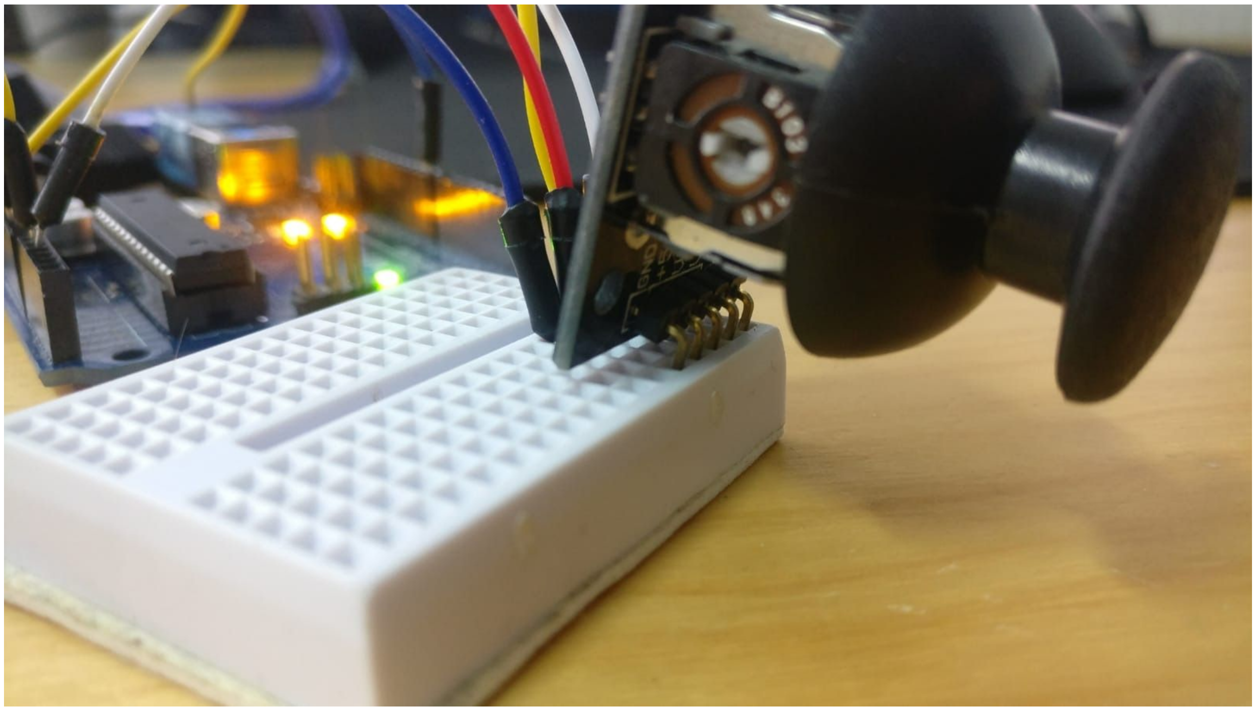
2.2 Reading values from Joystick

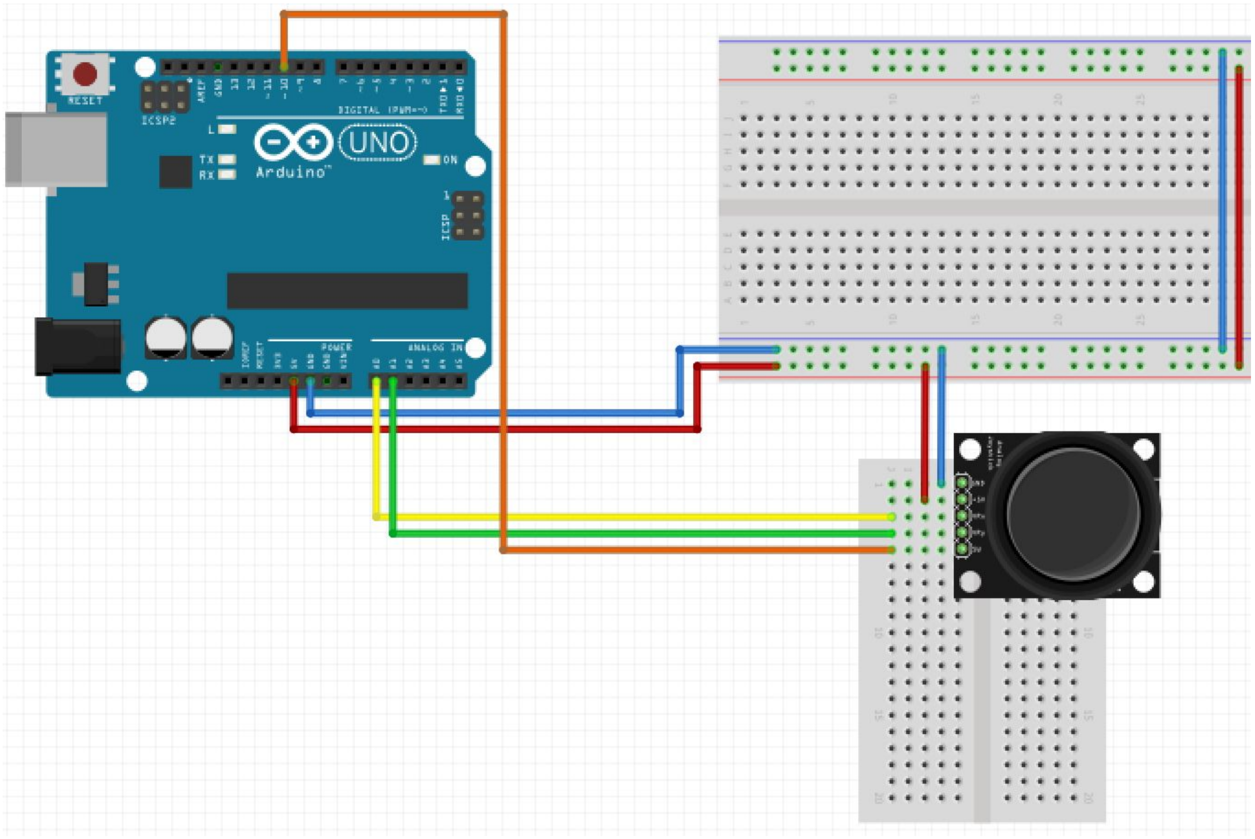
Questions:

1. Now we know we have 2 potentiometers and a pushbutton. How many and what type of pins do we need?
    a. **A**: 2 analog pins for potentiometers and 1 digital pin for pushbutton
2. We have analog data processed by digital processor (microcontroller). How is this possible?
    a. **A:** By using an intermediate device to convert the analog data into digital data, named ADC (Analog to Digital Converter)

In order to read the joystick's physical position, we need to measure the change in resistance of a potentiometer.

We'll use the joystick with the small breadboard so that we can add the other elements on the medium breadboard without removing the joystick. Make sure you connect your joystick in such a way that you can use it (don't put the wires in front of it).





```
// declare all the pins
const int pinSW = 10; // digital pin connected to switch output
const int pinX = A0; // A0 - analog pin connected to X output
const int pinY = A1; // A1 - analog pin connected to Y output
int switchValue;
int xValue = 0;
int yValue = 0;
void setup() {
  pinMode(pinSW, INPUT_PULLUP); //activate pull-up resistor on the
// push-button pin
  // Start the serial communication.
  Serial.begin(9600);
}

void loop() {
  switchValue = digitalRead(pinSW);
  xValue = analogRead(pinX);
  yValue = analogRead(pinY);
  Serial.print("Switch:  ");
  Serial.print(switchValue);
  Serial.print("  |  ");
  Serial.print("X-axis: ");
  Serial.print(xValue);
  Serial.print("  |  ");
  Serial.print("Y-axis: ");
  Serial.print(yValue);
  Serial.println("  |  ");
  delay(200);
}
```

Based on the values displayed on the serial monitor, try and decipher which direction is X and which direction is Y. If needed, change the delay value to a bigger one.

Questions:

1. What are the range values for Ox and Oy axis?
    a. **A**: The values on each analog channel (axis) can vary from 0 to 1023
2. When the joystick stays in its center position the value is around...?
    a. **A:** The value is around 512

2.2 Joystick & 7-segment display

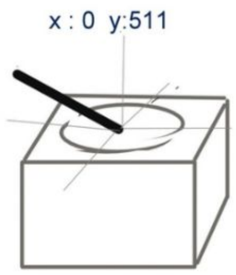
Now we know how to control a joystick and a 7-segment display individually. Let's use those components in the same system as follows:

- ★ 7-segment display: used for displaying numbers given as input though the variable named `digit`
- ★ Joystick: used for changing `digit` variable value by increasing with 1 unit if the joystick thumbstick is moved on the right side of the Ox axis (values bigger than 512) or decreasing with 1 unit if the joystick thumbstick is moved on the left side of the Ox axis (values lower than 512).
- ★ Use the joystick's button to toggle the decimal point on and off

Example:

Initial state: `xValue` = 512, `digit` = **2**. *We also have `yValue` = 511, but for this exercise won't be needed.*

Step 1: Move the thumb stick on the left side of the Ox axis and keep it in this position.



Result: `xValue` = 0, `digit` = **1**. As you can see, the `digit` is decremented with only 1 unit, even though we keep the joystick at 0 on Ox axis.

Step 2: Release the joystick thumb stick. Only the `xValue` value should change.

`xValue` = 512, `digit` = **1**.

Repeat step 1. Now the result should be

`xValue` = 0, `digit` = **0**.

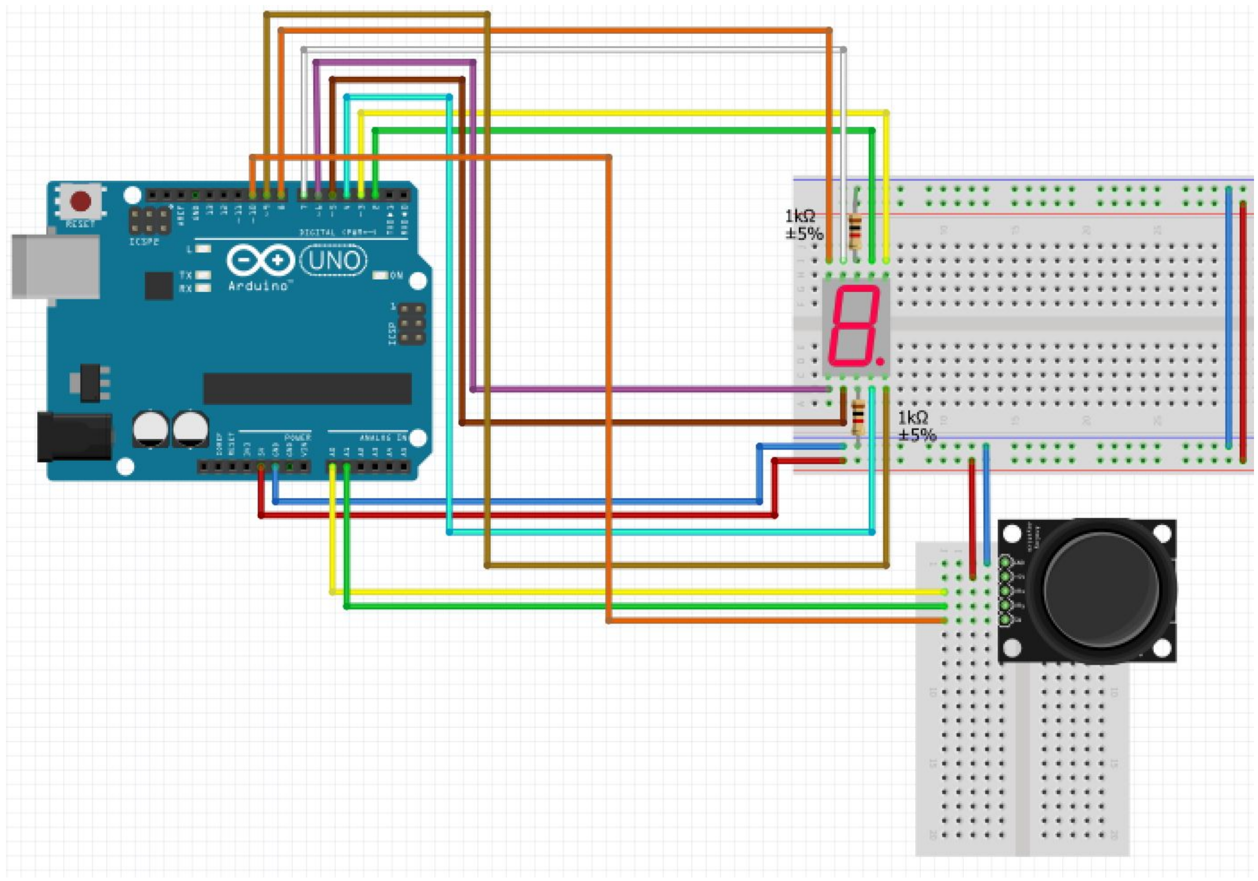
Release the joystick thumb stick and if we repeat again de step 1, the result should be

`xValue` = 0, `digit` = **9**.

Questions:

1. What digit will be displayed if `digit` = **9** and we increment it with 1 unit?

   A: `digit` = **0**.



```
// declare all the joystick pins
const int pinSW = 10; // digital pin connected to switch output
const int pinX = A0; // A0 - analog pin connected to X output
const int pinY = A1; // A1 - analog pin connected to Y output

// declare all the segments pins
const int pinA = 2;
const int pinB = 3;
const int pinC = 4;
const int pinD = 5;
const int pinE = 6;
const int pinF = 7;
const int pinG = 8;
const int pinDP = 9;

const int segSize = 8;
const int noOfDigits = 10;
```

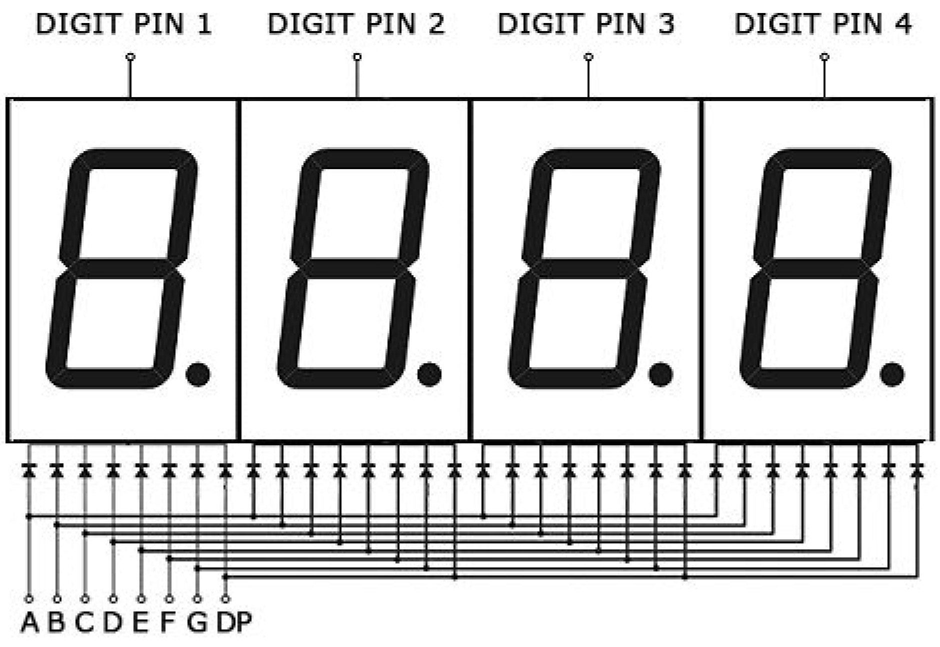
```
// dp on or off
int dpState = LOW;
// states of the button press
int swState = LOW;
int lastSwState = LOW;
int switchValue;
int xValue = 0;
int yValue = 0;

bool joyMoved = false;
int digit = 0;
int minThreshold= 400;
int maxThreshold= 600;

// store the pins in an array for easier access
int segments[segSize] = {
  pinA, pinB, pinC, pinD, pinE, pinF, pinG, pinDP
};

byte digitMatrix[noOfDigits][segSize - 1] = {
// a  b  c  d  e  f  g
  {1, 1, 1, 1, 1, 1, 0}, // 0
  {0, 1, 1, 0, 0, 0, 0}, // 1
  {1, 1, 0, 1, 1, 0, 1}, // 2
  {1, 1, 1, 1, 0, 0, 1}, // 3
  {0, 1, 1, 0, 0, 1, 1}, // 4
  {1, 0, 1, 1, 0, 1, 1}, // 5
  {1, 0, 1, 1, 1, 1, 1}, // 6
  {1, 1, 1, 0, 0, 0, 0}, // 7
  {1, 1, 1, 1, 1, 1, 1}, // 8
  {1, 1, 1, 1, 0, 1, 1}  // 9
};

void displayNumber(byte digit, byte decimalPoint) {
  for (int i = 0; i < segSize - 1; i++) {
      digitalWrite(segments[i], digitMatrix[digit][i]);
  }

  // write the decimalPoint to DP pin
  digitalWrite(segments[segSize - 1], decimalPoint);
}
```

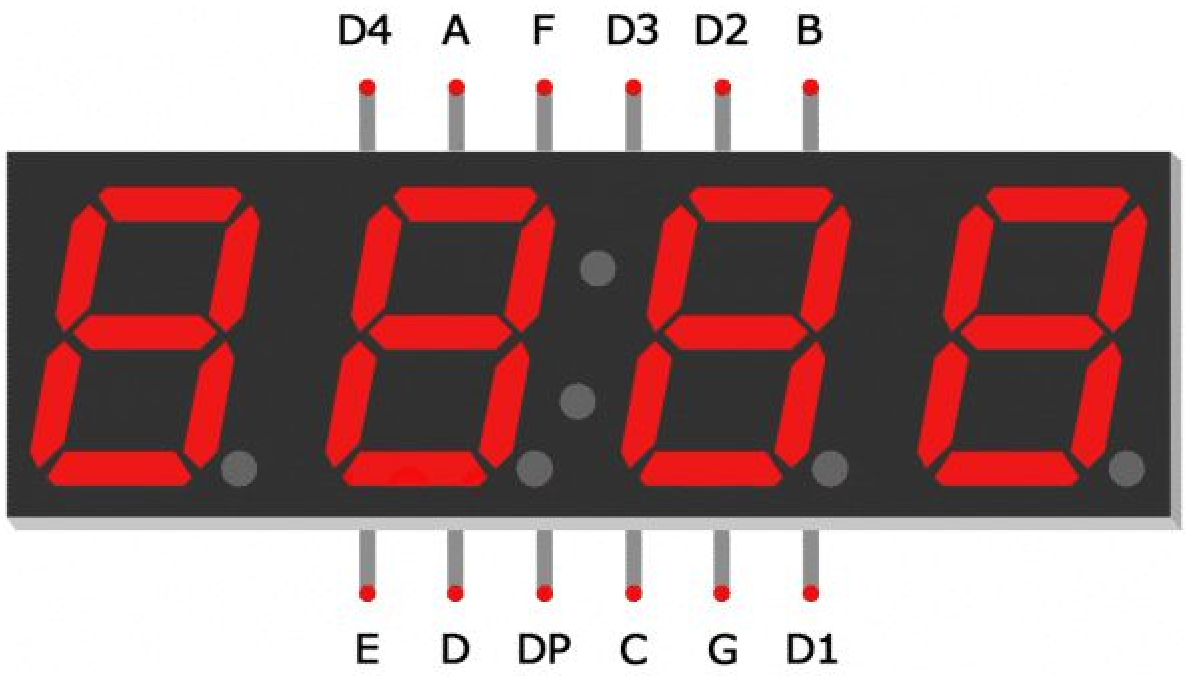
```
void setup() {
  // initialize all the pins
  for (int i = 0; i < segSize; i++) {
      pinMode(segments[i], OUTPUT);
  }
  pinMode(pinSW, INPUT_PULLUP);

  displayNumber(digit, dpState); // initial value displayed. Choose any
value
}

void loop() {
  xValue = analogRead(pinX);
 // On Ox axis, if the value is lower than a chosen min threshold, then
 // decrease by 1 the digit value.
  if (xValue < minThreshold && joyMoved == false) {
      if (digit > 0) {
        digit--;
      } else {
        digit = 9;
      }
      joyMoved = true;
  }

 // On Ox axis, if the value is bigger than a chosen max threshold, then
 // increase by 1 the digit value
  if (xValue > maxThreshold && joyMoved == false) {
      if (digit < 9) {
        digit++;
      } else {
        digit = 0;
      }
      joyMoved = true;
  }

  if (xValue >= minThreshold && xValue <= maxThreshold) {
      joyMoved = false;
  }

  // simple state change detector. Ideally, use debounce.
  swState = digitalRead(pinSW);
  if (swState !=  lastSwState) {
      if (swState == LOW) {
        dpState = !dpState;
      }
  }
  lastSwState = swState;

  displayNumber(digit, dpState);
  delay(1);
}
```

3. 4 digit 7-segment display

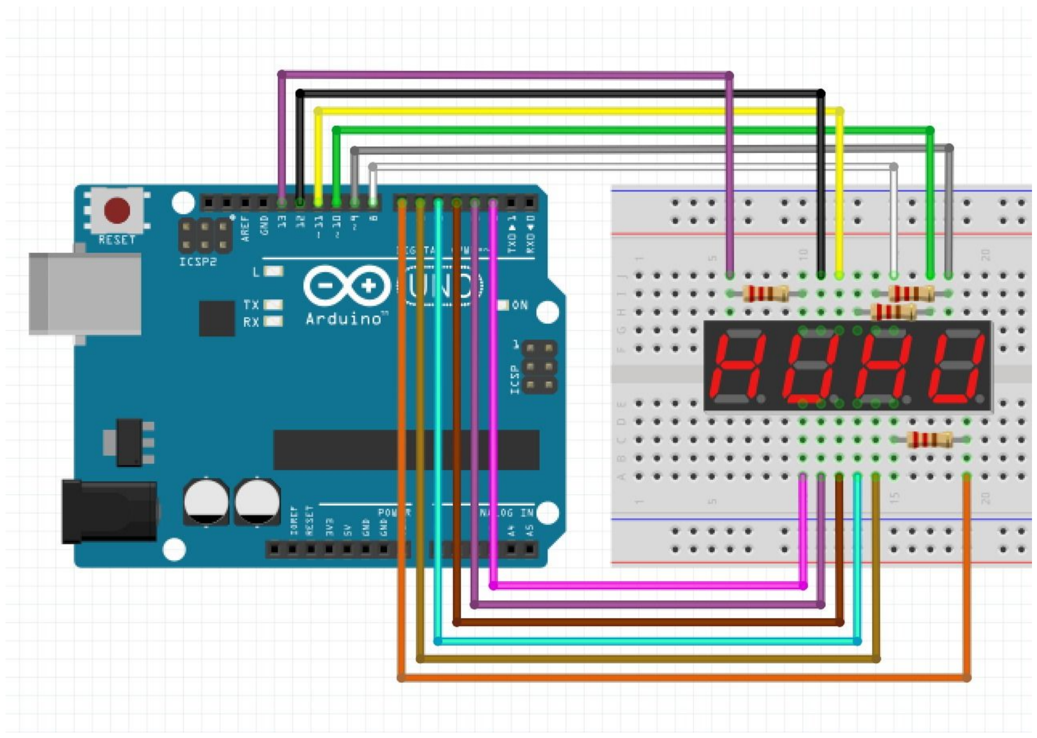
So far we have only worked with single digit 7-segment displays. To display information such as the time or temperature, you will want to use a 2 or 4 digit display, or connect multiple single digit displays side by side.

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/



```
const int pinA = 12;
const int pinB = 8;
const int pinC = 5;
const int pinD = 3;
const int pinE = 2;
const int pinF = 11;
const int pinG = 6;
const int pinDP = 4;
const int pinD1 = 7;
const int pinD2 = 9;
const int pinD3 = 10;
const int pinD4 = 13;

const int segSize = 8;

const int noOfDisplays = 4;
const int noOfDigits = 10;

int dpState = LOW;

int currentNumber = 0;
unsigned long delayCounting = 50;   // incrementing interval
unsigned long lastIncreasing = 0;
```

```
// segments array, similar to before
int segments[segSize] = {
  pinA, pinB, pinC, pinD, pinE, pinF, pinG, pinDP
};
// digits array, to switch between them easily
int digits[noOfDisplays] = {
  pinD1, pinD2, pinD3, pinD4
 };

byte digitMatrix[noOfDigits][segSize - 1] = {
// a  b  c  d  e  f  g
  {1, 1, 1, 1, 1, 1, 0}, // 0
  {0, 1, 1, 0, 0, 0, 0}, // 1
  {1, 1, 0, 1, 1, 0, 1}, // 2
  {1, 1, 1, 1, 0, 0, 1}, // 3
  {0, 1, 1, 0, 0, 1, 1}, // 4
  {1, 0, 1, 1, 0, 1, 1}, // 5
  {1, 0, 1, 1, 1, 1, 1}, // 6
  {1, 1, 1, 0, 0, 0, 0}, // 7
  {1, 1, 1, 1, 1, 1, 1}, // 8
  {1, 1, 1, 1, 0, 1, 1}  // 9
};


void displayNumber(byte digit, byte decimalPoint) {
  for (int i = 0; i < segSize - 1; i++) {
      digitalWrite(segments[i], digitMatrix[digit][i]);
  }

  // write the decimalPoint to DP pin
  digitalWrite(segments[segSize - 1], decimalPoint);
}

// activate the display no. received as param
void showDigit(int num) {
  for (int i = 0; i < noOfDisplays; i++) {
      digitalWrite(digits[i], HIGH);
  }
  digitalWrite(digits[num], LOW);
}
```

```
void setup() {
  for (int i = 0; i < segSize - 1; i++)
  {
      pinMode(segments[i], OUTPUT);
  }
  for (int i = 0; i < noOfDisplays; i++)
  {
      pinMode(digits[i], OUTPUT);
  }
  Serial.begin(9600);
}

void loop() {
  int number;
  int digit;
  int lastDigit;
  Serial.println(currentNumber);

  number = currentNumber;
  digit = 0;

  while (number != 0) {
      lastDigit = number % 10;  // get the last digit
      showDigit(digit);
      displayNumber(lastDigit, HIGH);
      // increase this delay to see multiplexing in action. At about 100 it
becomes obvious
      delay(5);
      digit++;                // move to next display
      number = number / 10;
  }
  // increment the number
  if (millis() - lastIncreasing >= delayCounting) {
      currentNumber = (currentNumber + 1) % 10000;
      lastIncreasing = millis();
  }
}
```

5. Homework

1. Deadline: ~~Your laboratory in week **18th - 22nd of November**~~ **Preferably at the lab in week 18th - 22nd, but due to the lab being made available late, you can present it anytime during the week at room 314. Write a message in advance to make sure someone is there, but you are responsible for presenting during this week.**

2. Minimal required items:

a. 1 * 4 digit 7-segment display
b. 1 * joystick
c. Resistors
d. Wires
e. Arduino
f. Connector cable

3. Task: Individually controlling each digit of the segment with the joystick

a. Using the X-axis, cycle through the digits and chose which one to modify. Light up the DP led of the current position
b. Press the button to lock on the selected digit
c. After that, use the Y-axis to modify the value of the digit, both up and down.
d. Press the button again to unlock the selection and be able to cycle through the 4 digits again

4. Notes:

a. The X and Y axis must be used exclusively. When you cycle through the digits, you cannot modify the value. When you are locked in on a digit, you cannot switch to another one without pressing the button

5. Extra points:

a. There are 3 states: locked in / selected (not locked in though) / not selected. With the DP you can only show 2 states. Differentiate, in a way, between all 3. One way to do it is with blinking digit or decimal point.