

Introduction to robotics

4th 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.

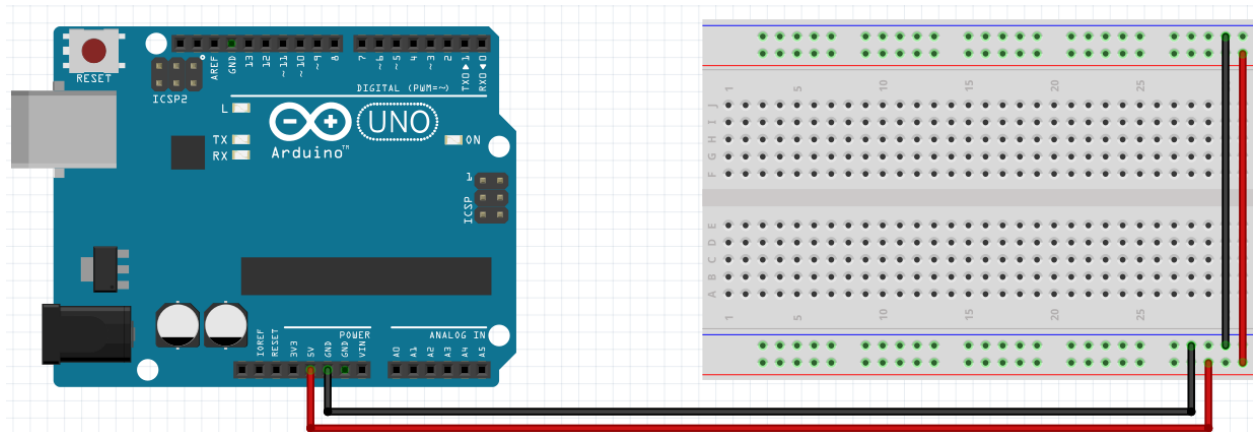


Fig. 1.1 - Default setup

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:

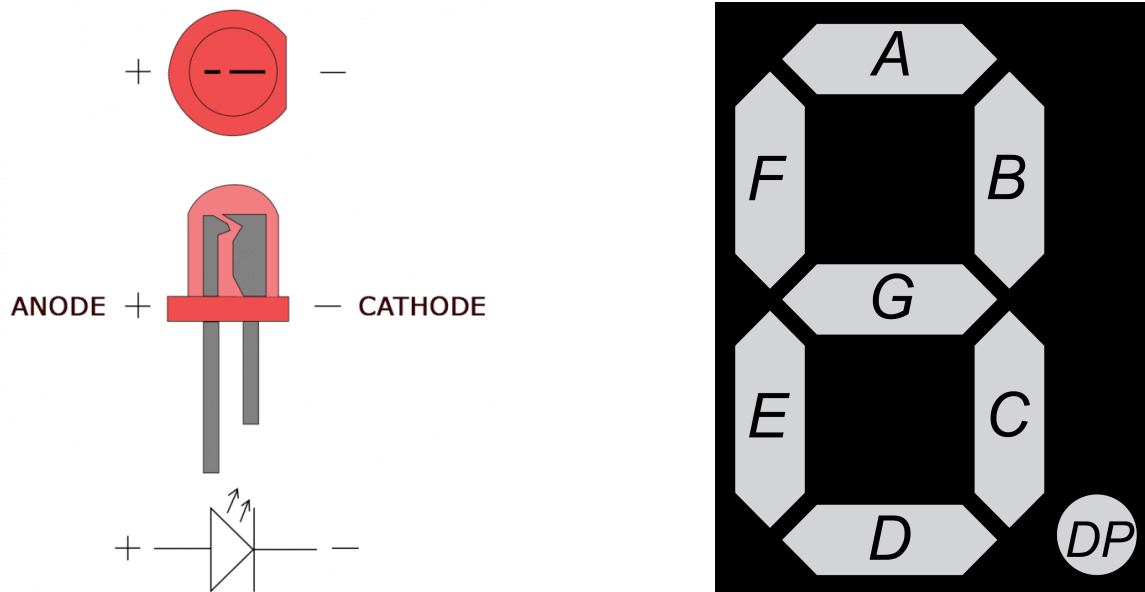


Fig. 1.2 - Light Emitting Diode (left), 7-segment-display LEDs encoding (right)
(source: https://en.wikipedia.org/wiki/Seven-segment_display)

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 a common cathode, it's best to **check it with a multimeter**.

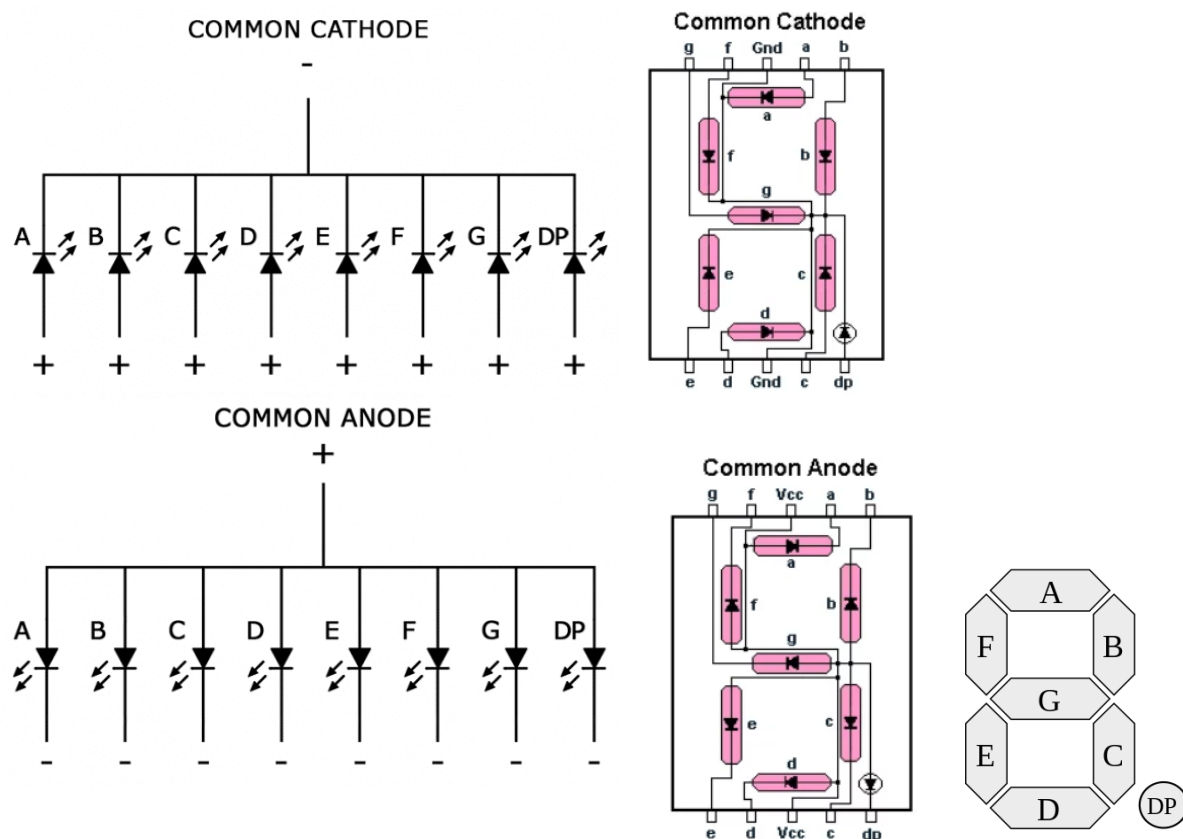


Fig. 1.3 - 7 segment display setup: Common Cathode (top) vs Common Anode (bottom)

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

Fig. 1.4 - Encodings combinations for punctuations (left) and digits 0 to F (right)
(source: https://en.wikipedia.org/wiki/Seven-segment_display).

Example of short messages:

OPEN, CLOSE, PLAY, PAUSE, SHUFFLE, NO DISC
 SEARCH, STOP, RUN, FASE, ERROR, SETUP, HELP
 ON, OFF, YES, NO, HOT, COLD

Fig. 1.5 - Different messages that can be written on 4 digit 7 segment display

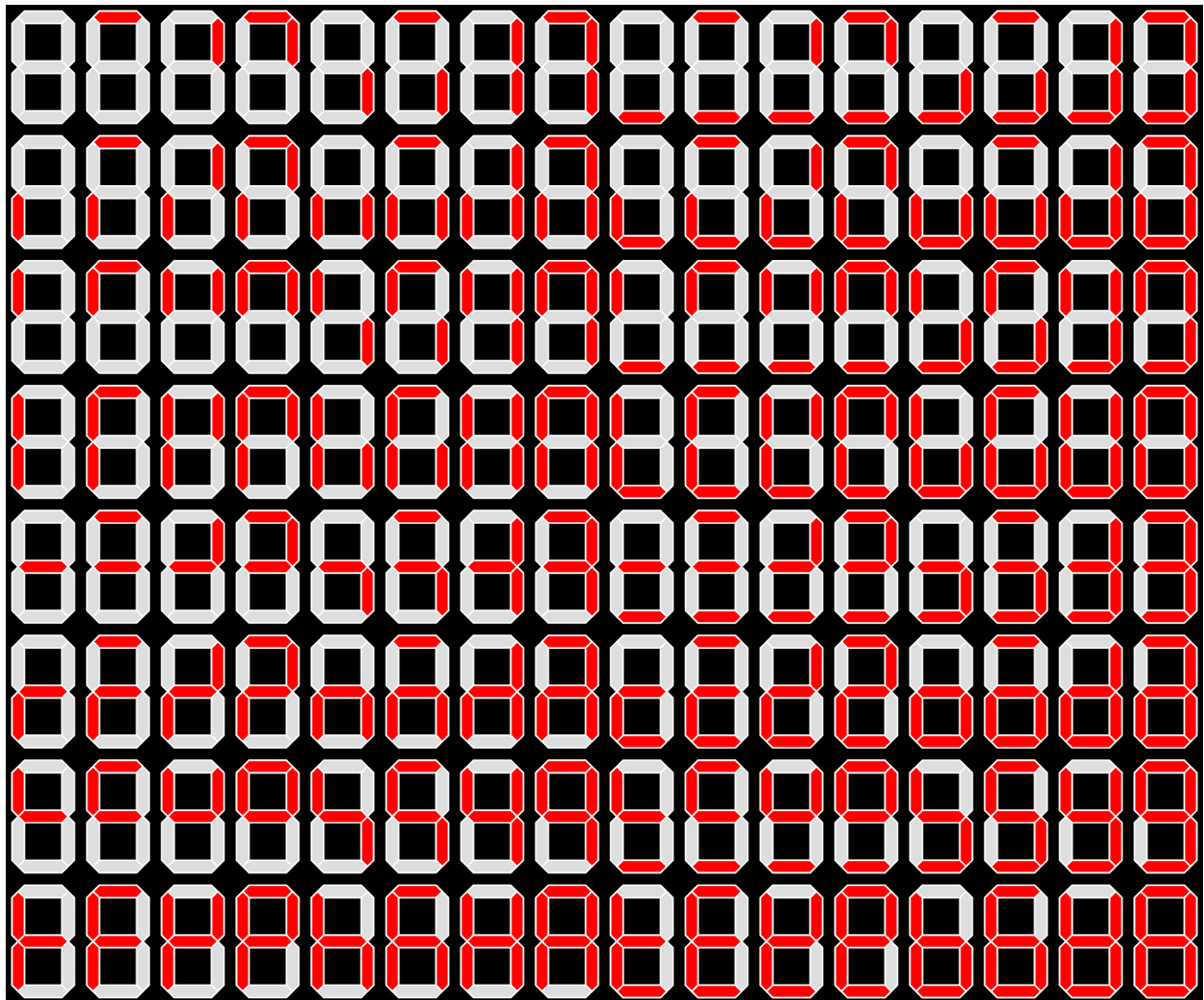


Fig. 1.6 - 16x8-grid showing the 128 states of a seven-segment display

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 start 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 a 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.

1.4.1 Connecting the 7-segment display - common cathode

For common anode, check Chapter 5. Extra content

Arduino PIN	Display PIN	Schematic (common Cathode)
4	A (7)	
5	B (6)	
6	C (4)	
7	D (2)	
8	E (1)	
9	F (9)	
10	G (10)	
11	DP (5)	
GND	3, 8 (GND)	Through a 1k+ Ω resistor

Table 1 - Arduino connections to 7 segment display **common cathode**

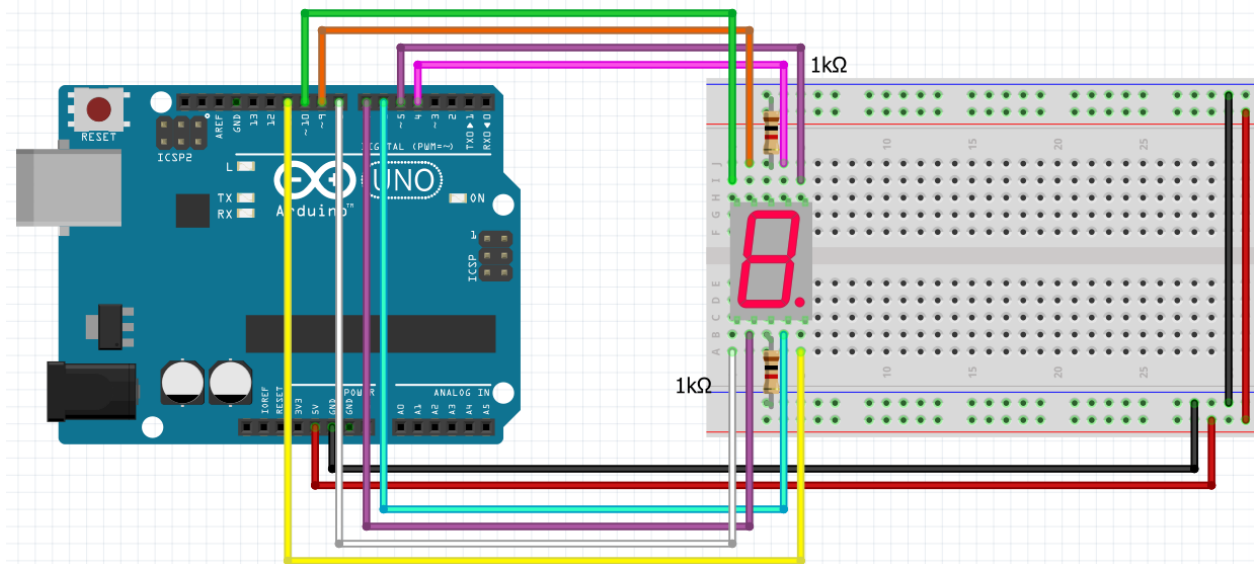


Fig. 1.7 - Arduino connections to 7 segment display **common cathode**

1.5 Checking each segment

```
// declare all the segments pins
const int pinA = 4;
const int pinB = 5;
const int pinC = 6;
const int pinD = 7;
const int pinE = 8;
const int pinF = 9;
const int pinG = 10;
const int pinDP = 11;

const int segSize = 8;
int index = 0;
// modify if you have common anode
bool commonAnode = false;

byte state = HIGH;
int segments[segSize] = {
    pinA, pinB, pinC, pinD, pinE, pinF, pinG, pinDP
};

void setup() {
    // initialize all the pins
    for (int i = 0; i < segSize; i++) {
// TODO
    }
    if (commonAnode == true) {
        state = !state;
    }
}

void loop() {
    // TODO: cycle through all the segments, lighting them up and then
    // shutting them off. It is a good exercise to see if the pins are connected
    // properly
}
```

Code snippet 1.1 - Turning all the segments on, and then all the segments off, in order

1.6 Counting from 0 to 9

```
// declare all the segments pins
const int pinA = 4;
const int pinB = 5;
const int pinC = 6;
const int pinD = 7;
const int pinE = 8;
const int pinF = 9;
const int pinG = 10;
const int pinDP = 11;

const int segSize = 8;
int index = 0;

// modify if you have common anode
bool commonAnode = false;

const int noOfDigits = 10;
byte state = HIGH;
byte dpState = LOW;

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

void loop() {
    // TODO: Display the digits from 0 to 9 without dp and then from 0 to 9
    with dp
}

void displayNumber(byte digit, byte decimalPoint) {
    // TODO: write the commands so that the shows the input number
    // TODO: write the decimalPoint to DP pin
}
```

Code snippet 1.2 - Displaying, in order, digit 0 to 9 without decimal point and then with decimal point

2. Joystick

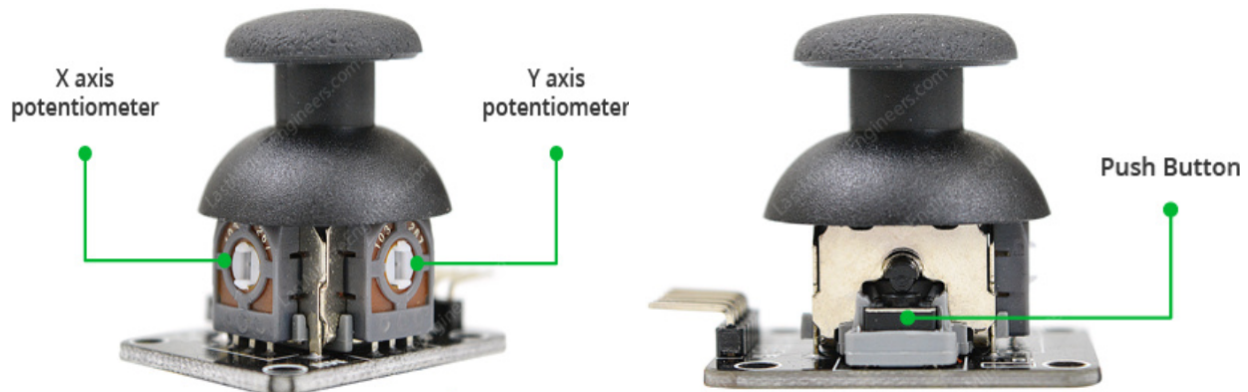


Fig. 2.1 - Joystick axis and button

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

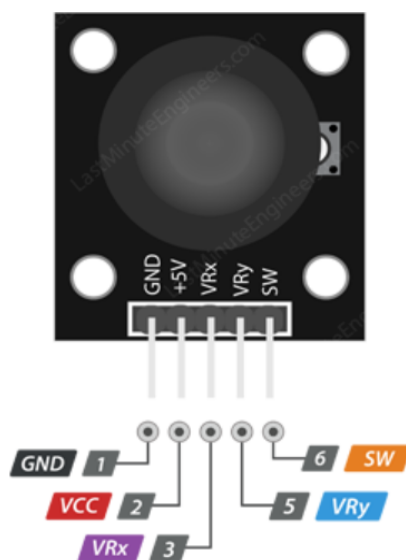


Fig 2.2 - Joystick pinout configuration

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.

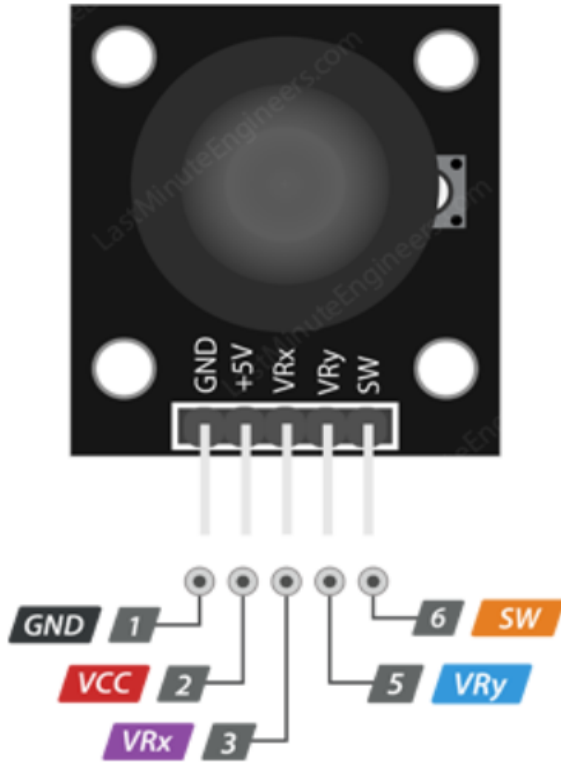
Arduino PIN	Joystick PIN	Schematic
GND	1 (GND)	
5V	2 (5V)	
A0	3 (VRx)	
A1	4 (VRy)	
2	5 (SW)	

Table 2.1 - Arduino connections to joystick

2.2 Reading values from Joystick

Questions:

- Now we know we have 2 potentiometers and a pushbutton. How many and what type of pins do we need?
 - A:** 2 analog pins for potentiometers and 1 digital pin for pushbutton
- We have analog data processed by a digital processor (microcontroller). How is this possible?
 - 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).

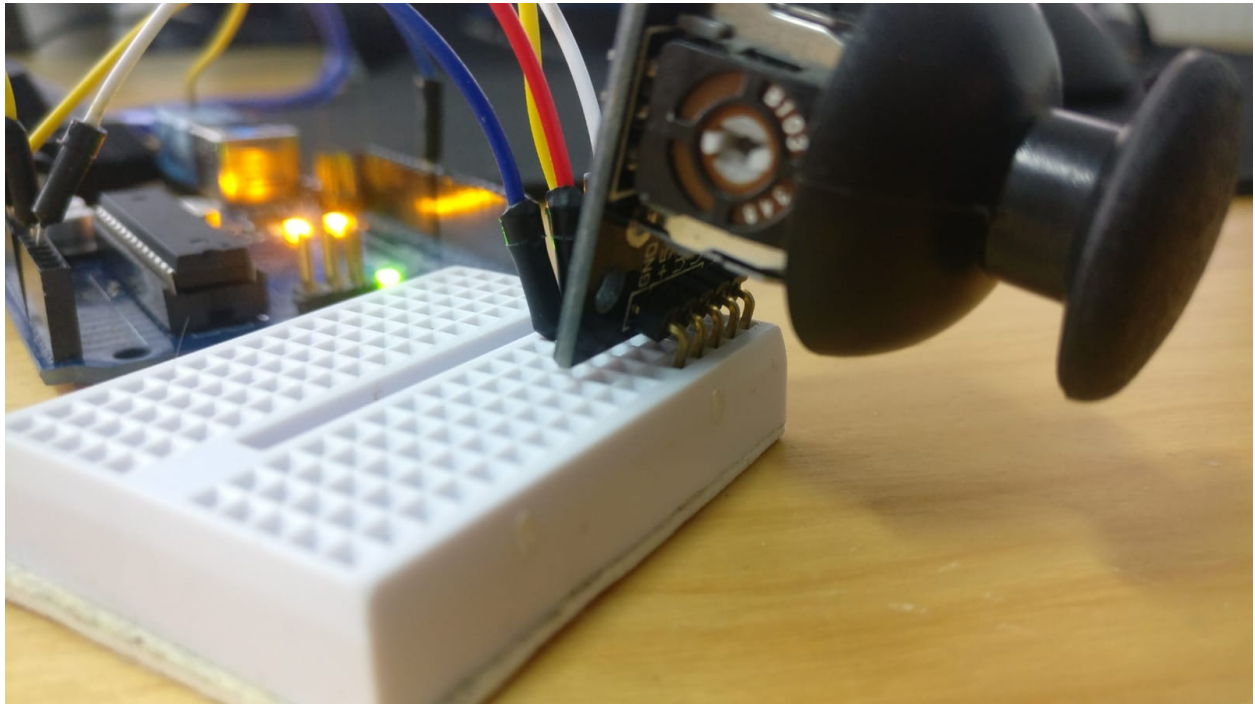


Fig. 2.2 - Recommended joystick positioning (on the small breadboard)

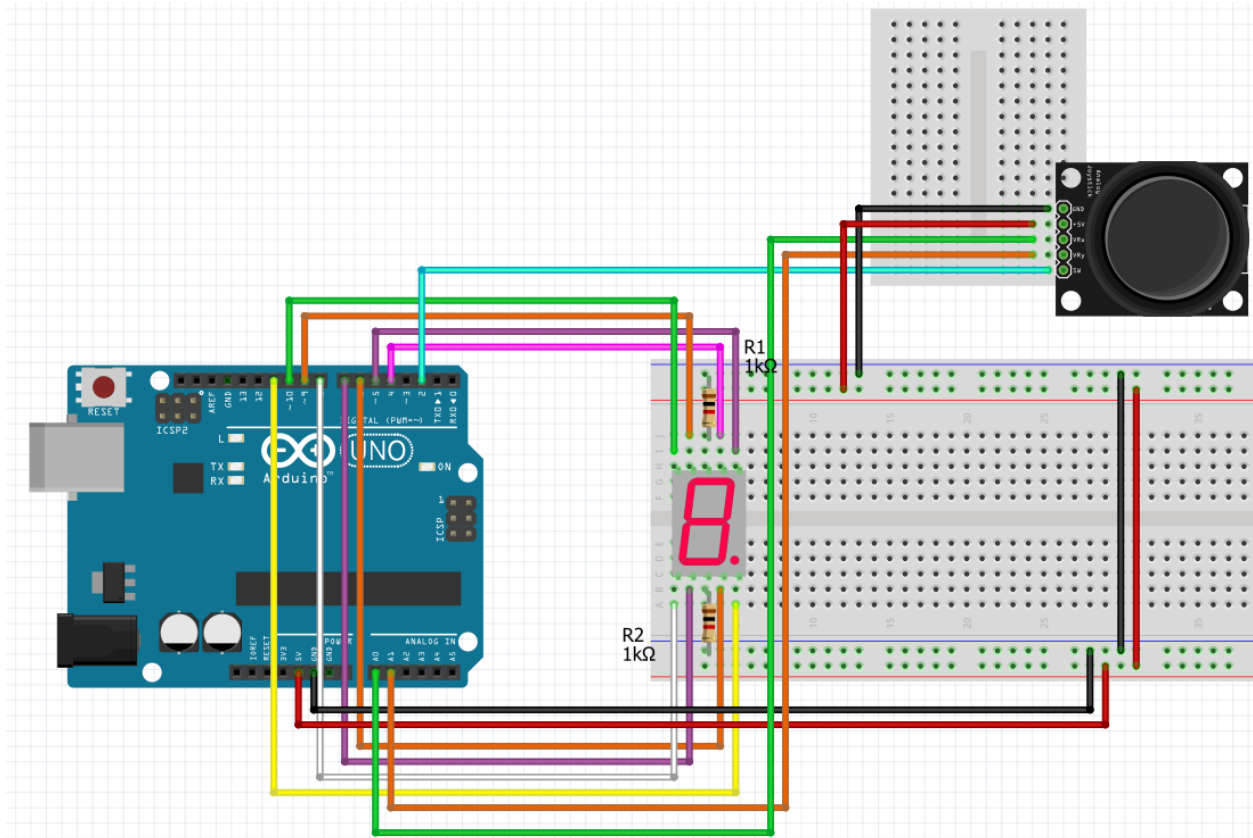


Fig. 2.3 - Joystick 7 segment display (common cathode) connections

```
// declare all the pins
const int pinSW = 2; // 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
byte swState = LOW;
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() {
    // TODO read joystick values
    swState =
    xValue =
    yValue =
```

```
Serial.print("Switch: ");  
Serial.print(swState);  
Serial.print(" | ");  
Serial.print("X-axis: ");  
Serial.print(xValue);  
Serial.print(" | ");  
Serial.print("Y-axis: ");  
Serial.print(yValue);  
Serial.println(" | ");  
delay(200);  
}
```

Code snippet 2.1 - Joystick interaction values

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 the Ox and Oy axis?
 - a. **A:**
2. When the joystick stays in its center position the value is around ... ?
 - a. **A:**

3. 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 through the variable named **digit**
- Joystick: used for changing **digit** variable

Details:

- Digit value increases by 1 if the joystick thumbstick is moved on the right side of the Ox axis (values bigger than 512) or decreases by 1 unit if the joystick thumbstick is moved on the left side of the Ox axis (values lower than 512)
- It should only move once with each iteration meaning that it doesn't continuously decrease if you keep the joystick moved to the right; you have to bring it to the middle and move it again if you want to decrease it further
- The joystick should use the joystick's button to toggle the decimal point on and off

```
// declare all the joystick pins
const int pinSW = 2; // 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 = 4;
const int pinB = 5;
const int pinC = 6;
const int pinD = 7;
const int pinE = 8;
const int pinF = 9;
const int pinG = 10;
const int pinDP = 11;

const int segSize = 8;
int index = 0;

// modify if you have common anode
bool commonAnode = false;

const int noOfDigits = 10;
byte state = HIGH;
byte dpState = LOW;
byte swState = LOW;
```



```

byte lastSwState = LOW;
int xValue = 0;
int yValue = 0;

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

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 setup() {
    // initialize all the pins
    for (int i = 0; i < segSize; i++) {
        pinMode(segments[i], OUTPUT);
    }
    pinMode(pinSW, INPUT_PULLUP);
    if (commonAnode == true) {
        state = !state;
    }
}

void loop() {
    // TODO: use joystick movement to cycle through the numbers from 0 to 9,
    with a state changer (aka if you move the joystick, only one increment
    should happen. The next change should be after the joystick went back to

```

```
initial position and is moved again).  
}  
  
void displayNumber(byte digit, byte decimalPoint) {  
    // TODO: same function from previous exercise  
}
```

Code snippet 3.1 - Incrementing (and decrementing) a digit displayed on the 7 segment display based on joystick movement, with joystick state

4. 4 digit 7 segment display (optional, home ex.)

// will be added soon

5. Extra content

5.1 Arduino connections table and figure to 7 segment display

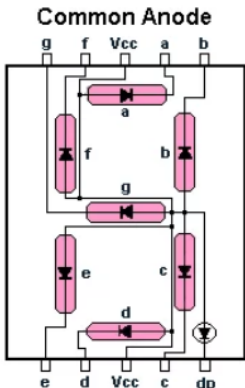
Arduino PIN	Display PIN	Schematic (common Anode)
4	A (7)	
5	B (6)	
6	C (4)	
7	D (2)	
8	E (1)	
9	F (9)	
10	G (10)	
11	DP (5)	
5V	3, 8 (VCC)	Through a 1k+ Ω resistor

Table 5.1 - Arduino connections to 7 segment display **common anode**

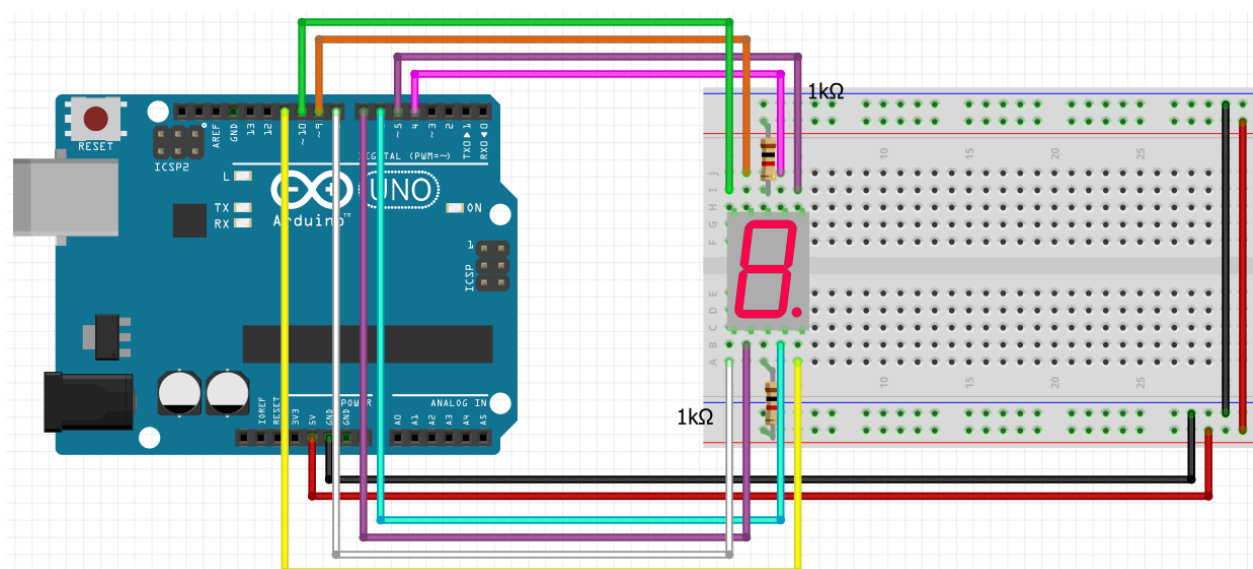


Fig. 5.1 - Arduino connections to 7 segment display **common anode**

5.2 Arduino connections to joystick and 7 segment display with common anode

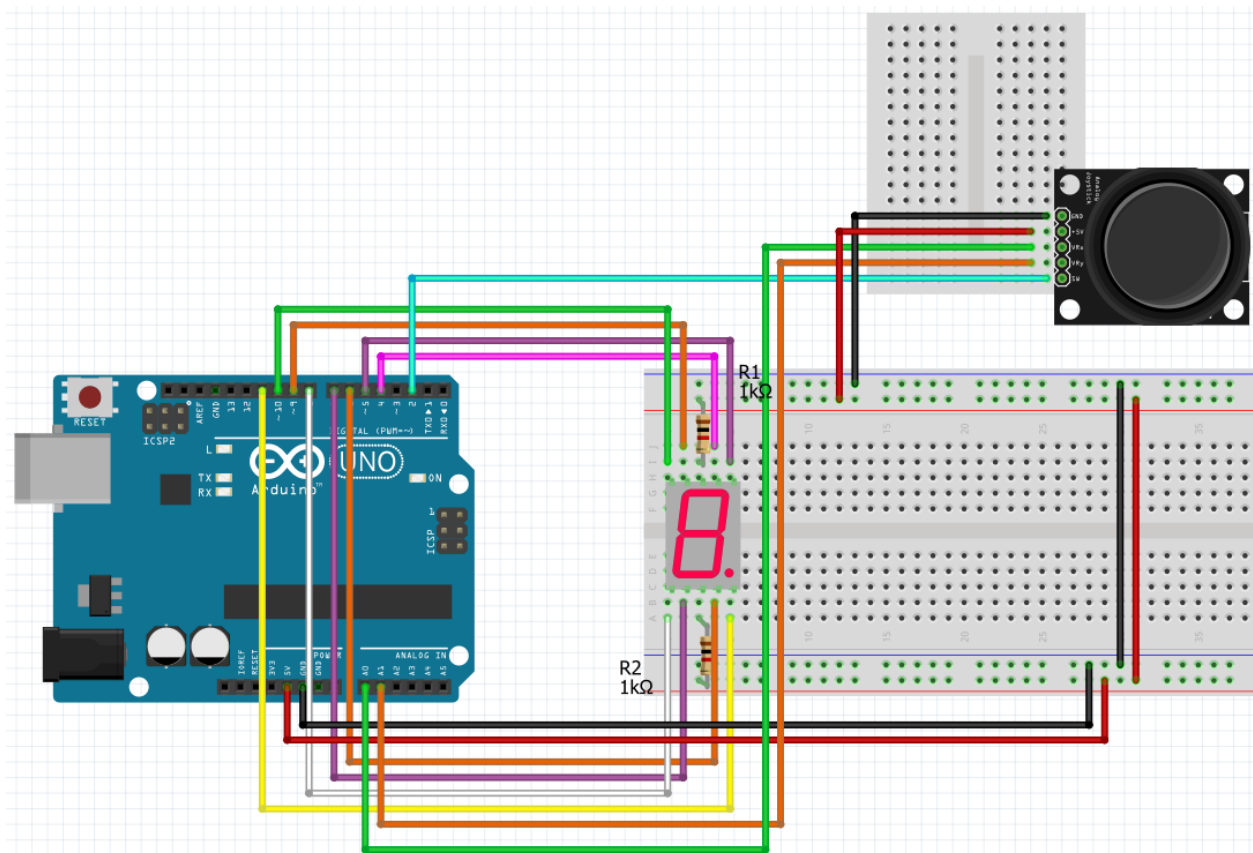


Fig. 5.2 - Joystick 7 segment display (**common cathode**) connections