# Arduino Lab 3 - Implement a Click-Counter

In this lab you will be implementing a basic click-counter. Basically, you will be using a variable to hold a count. On each button press the counter will be going up by values of 1. The count will be displayed for the user.

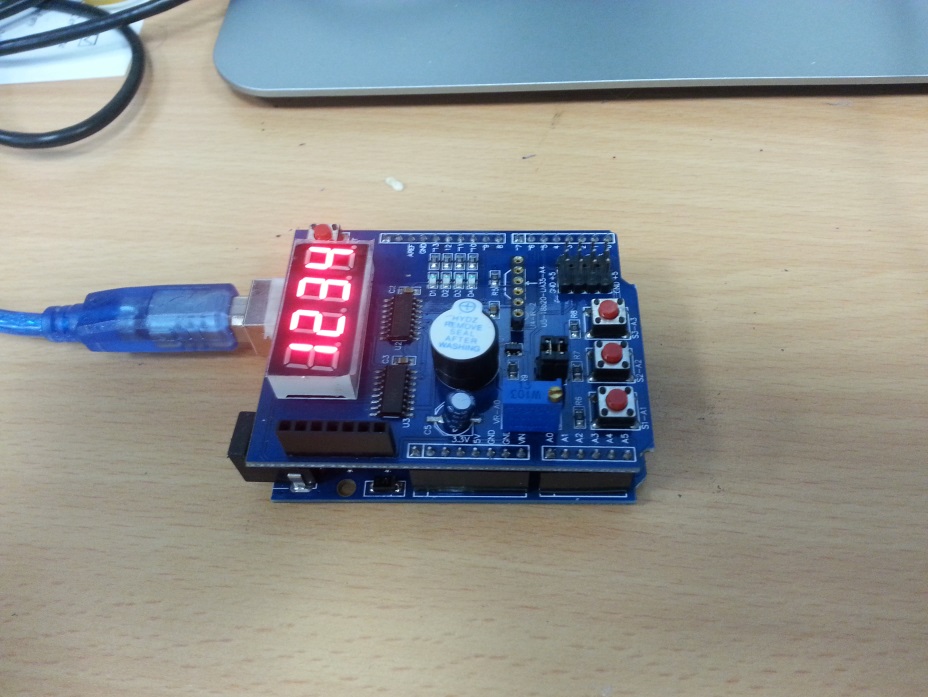
You will be using the Multi-Function shield (attaching it to the Ardiono). This device has both a seven-segment display (for the user to visualize the count) and several buttons, one of which you will use to increment the counter.

The steps to completing this lab are:

1. Read the tutorials on how to use the Arduino Multi-Function Shield
2. Attempt the lab by:
   1. Creating a counter variable
   2. Incrementing it on button presses
   3. Create a function to output the count to the seven-segment display

## Tutorials on: The Digital Multi-Function Shield

In this tutorial we will be using the digital multi-function shield. The wiring involves a simple attachment similar to the lcd screen (shown below). This device is capable of driving a 7-segment display, detecting button pressed (there are 3 buttons) and driving 4 LEDs on the shield.



### LED control

const byte LED[] = {13,12,11,10};

int highlightedPin = 0;

void setup()

{

// initialize the digital pin as an output.

/\* Set each pin to outputs \*/

for(int i = 0; i < 4; i++) pinMode(LED[i], OUTPUT);

}

void loop()

{

for(int i = 0; i < 4; i++)

if(i == highlightedPin) digitalWrite(LED[i], LOW);

else digitalWrite(LED[i], HIGH);

highlightedPin = (highlightedPin + 1) % 4;

delay(500);

}

In this first tutorial, we will be creating a small animation among the 4 leds on the shield. The digital output pins used to drive these LEDs are 13, 12, 11 and 10. Notice in out code, we have a highlighted pin variable as well as an LED byte array for indexing out pins. The highlightedPin variable can be 0, 1, 2 or 3 to turn on the corresponding LED. In our setup function we use the pinMode function to setup each pin as an output pin using a for loop. In our loop function, we have another for loop, and turn pin i on if it is equal to our highlightedPin otherwise we turn it off. We then set highlightedPin to the next index, looping around back to 0 if need be.

### Button Click Detection

To detect button clicks on the multi-shield, we have to read a digital input from Analoge pins A1, A2 or A3 (1 for each button). A digital read of 0 means the button is pressed, a digital read of 1 or HIGH means the button is not pressed.

In the code below, we setup out button pins as inputs, and our LED pins as outputs. If digitalRead(BUTTON1) returns LOW, we turn all our pins on, if we detect BUTTON2 was pressed, we turn all the pins off.

const byte LED[] = {13,12,11,10};

#define BUTTON1 A1

#define BUTTON2 A2

#define BUTTON3 A3

void setup()

{

// initialize the digital pin as an output.

/\* Set each pin to outputs \*/

for(int i = 0; i < 4; i++) pinMode(LED[i], OUTPUT);

pinMode(BUTTON1, INPUT);

pinMode(BUTTON2, INPUT);

pinMode(BUTTON3, INPUT);

}

void loop()

{

if(!digitalRead(BUTTON1))

for(int i = 0; i < 4; i++) digitalWrite(LED[i], HIGH);

if(!digitalRead(BUTTON2))

for(int i = 0; i < 4; i++) digitalWrite(LED[i], LOW);

}

### Driving the 7-segment display

In this example we will be using a new function called shiftOut(dataPin, clockPin, bit order, byte). This function writes a whole byte of data out on pin: dataPin. Below is a table indicating what each parameter means.

|  |  |  |
| --- | --- | --- |
| Index | Name | Description |
| 0 | dataPin | The pin in which data is sent out on |
| 1 | clockPin | The pin in which a clock signal is used, when we send data serially in electronics we usually use one pin (the clock) to signal a new bit is ready to be read and another pin (data) to indicate which bit should be read (1 or 0). |
| 2 | Bit order | Can be MSBFIRST or LSBFIRST indicating that the most significant bit should be written first or the least significant bit, this decides the order in which to send the 8 bits of out byte |
| 3 | byte | The byte to send |

Next comes the code:

/\* Define shift register pins used for seven segment display \*/

int latch = 4;

int clk = 7;

int data = 8;

/\* bytes which id one of the 4 7 segment displays \*/

const byte segmentIds[] = {0xF1,0xF2,0xF4,0xF8};

void setup ()

{

/\* Set DIO pins to outputs \*/

pinMode(latch,OUTPUT);

pinMode(clk,OUTPUT);

pinMode(data,OUTPUT);

}

void drive7Segment(byte segment, int id){

digitalWrite(latch,LOW);

shiftOut(data, clk, MSBFIRST, segment);

shiftOut(data, clk, MSBFIRST, segmentIds[id] );

digitalWrite(latch,HIGH);

}

/\* Main program \*/

void loop()

{

/\*write a number to each 7 segment\*/

drive7Segment(~0x06, 0);

drive7Segment(~0x5B, 1);

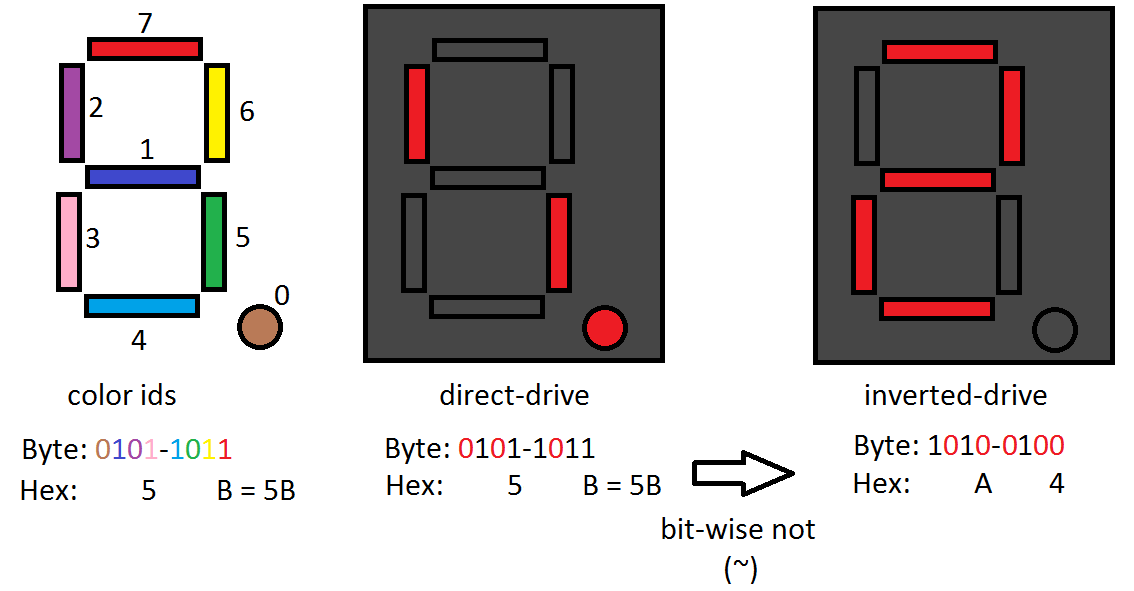
drive7Segment(~0x4F, 2);

drive7Segment(~0x66, 3);

}

Here we define some pins in global space latch, data and clk. The data and clk (clock) pins are set as pins 8 and 7, and will be used in our shiftOut function for sending bytes to the shield. The latch pin lets the shield know whether serial data is coming in. In our setup function we setup these pins as outputs.

In our main loop, we are using a function drive7Segment which takes two parameters, the byte to drive a seven segment display (explained soon). And an index into global array segmentIds which contains the id codes for each 7 segment display (1 of 4). Now, for the first parameter which drives a seven-segment display is a representation in binary for which segments we want turned on/off. Take a look at the picture below:



We can see that for a given byte (8 bits) we can align those bits up (by colour in the picture) to one of the 7 segments, for example say we want to drive the number 2. We could take a look at which segments we want to be ON and which OFF. For the number 2, the on segments should be (dark blue, pink, light blue, yellow and red). If we turn those bits on, then we have the result in the middle picture (direct-drive). This is the opposite of what we want, on the multi-shield, the 7-segment display byte signal should contain a 0 at a particular bit location to turn the segment ON. If we invert out bits 0101-1011 (hex: 0x5B) to 1010-0100 (hex: 0xA4), we have the correct representation for 2.

Back to our code, when we write drive7Segment(~0x06, 1), we are taking 0x06 (0000-0110) and inverting it becomes 0xF9 (1111-1001) which is the correct drive for the number 1. See if you can figure out what the rest of the loop function does based on this pattern.

Finally we have our drive7Segment function, taking the byte segment pattern (discussed above) and the index to the segmentIds array to indicate which of the 4 7 segment displays we wish to alter. Before we send the shield any information we must set the latch output to LOW. After communication we set it high again. We then use the shiftOut function to send two bytes of data (uses 2 calls 1 for each byte). The first is our segment pattern, the second is the id of the 7-segment we wish to display.

See if you can program your own driver to display all of the Hex values: 0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f.

# Lab 3: Click Counter

If you have not yet read the introduction to the 7-Segment Display in the Digital Multi-Function Shield section and the Button Click detection in the same section. In this labe you will be required to implement a counter with a button click as input. Essentially, the seven segment display will display the value of a counter, the counter will go from 0 to 100 then loop back to 0. Choose a button on the Seven Segment Display to act as an event generator for the counter to increase.

Use a variable "count" which has an initial value of zero, and a function called mustIncrement() which is called in a loop. When the button is pressed it returns a 1 else 0. Use this to increment count. Remember the seven-segment display must be driven using a look-up table and you must find each decimal place value in "count." For example the number 64 has 6 in the tens location and a 4 in the ones location. Use integer division ( / ) and the modulo function ( % ) to find these values.

Write a function showNumber(); which shows the count variable in the seven segment display.

Your loop function may therefore look like the following:

void loop()

{

count += checkIncrement();

showNumber(count);

}