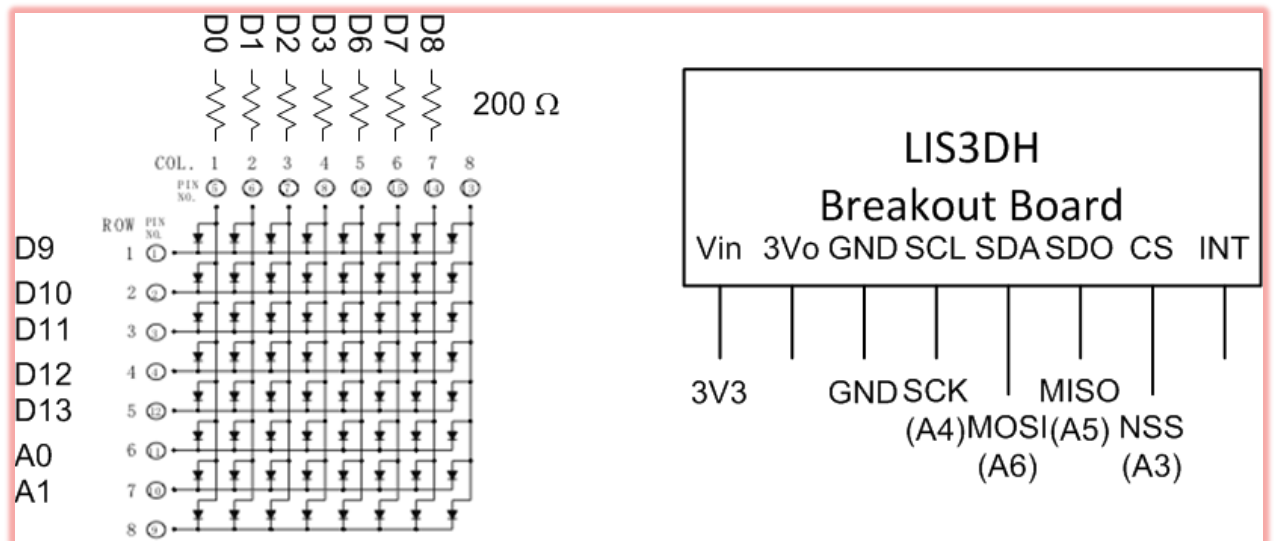


Lab 9: Digital Level SOLUTIONS

Digital Design and Computer Architecture: RISC-V Edition (Harris & Harris, Elsevier © 2021)

1. Please indicate how many hours you spent on this lab. This will be helpful for calibrating the workload for next time the course is taught.
2. [1] Schematic of the circuit on your breadboard, showing all components and which Nucleo pins are connected.



Anything between 100 Ω and 330 Ω will work for the resistors. They can be placed either on the rows or columns, but not both. A0 and A1 are also D14 and D15.

3. [3] C code for your digital level.

```
// E85 Lab 8: Digital Level
// Cherie Ho and David Money Harris
// with minor additions by erik_spjut@hmc.edu 2017-11-09
// Based on Bryce's SPI + GPIO STM32F0 CMSIS example

#include "stm32f0xx.h"
#include "EasyNucleoIO.h"

// mapping of rows and columns to digital pins
// don't use pins D4 or D5 because they are bridged for other purposes
int colPins[7] = {0, 1, 2, 3, 6, 7, 8};
int rowPins[7] = {9, 10, 11, 12, 13, 14, 15}; // Pins D14 & D15 are also A0 and A1

// SPI Functions

#define ALT 2

void spilnit() {
    //set ALT function for SPI pins
    pinMode(17, ALT); // PA4/SPI1_NSS (A3)
    pinMode(18, ALT); // PA5/SPI1_SCK (A4)
    pinMode(19, ALT); // PA6/SPI1_MISO (A5)
    pinMode(20, ALT); // PA7/SPI1_MOSI (A6)

    //configure SPI:
    //SPI enabled -> SPE; BAUD rate -> BR; Master -> MSTR;
    //BR = fclk/8, master mode
}
```

```

    SPI1->CR1 |= SPI_CR1_BR_1 | SPI_CR1_MSTR;
    // Enable NSS select signal, pulse NSS between transfers, 16-bit data,
    SPI1->CR2 |= SPI_CR2_SSOE | SPI_CR2_NSSP | (0xF) << 8;
    //enable SPI
    SPI1->CR1 |= SPI_CR1_SPE;
}

//send a 16-bits (short) on SPI1
unsigned short SPI1SendReceive16(unsigned short outDat) {
    SPI1->DR = outDat;
    while(!(SPI1->SR & SPI_SR_RXNE)); // wait for response
    return SPI1->DR;
}

void spiWrite(unsigned char address, unsigned char value) {
    SPI1SendReceive16(address << 8 | value);
}

unsigned char spiRead(unsigned char address) {
    return SPI1SendReceive16(address << 8 | (1 << 15));
}

////////////////////////////////////
// Display Functions
////////////////////////////////////

void GPIOInit(void){
    int i;

    for (i=0; i<7; i++) {
        pinMode(rowPins[i], OUTPUT);
        pinMode(colPins[i], OUTPUT);
    }
}

void clearGPIO(){
    // set anodes to 0 and cathodes to 1 to turn off all LEDs
    int i;

    for (i=0; i<7; i++) {
        digitalWrite(colPins[i], 0); // anodes
        digitalWrite(rowPins[i], 1); // cathodes
    }
}

void driveLED(int row, int col){
    // turn off all dots
    clearGPIO();

    // turn on desired row and column
    digitalWrite(colPins[col], 1);
    digitalWrite(rowPins[row], 0);
}

////////////////////////////////////
// Digital Level logic
////////////////////////////////////

int clip(int val, int max) {
    if (val < -max) return -max;
    else if (val > max) return max;
    else return val;
}

int main(void)
{
    volatile unsigned char debug;
    volatile short x,y, disx,disy;

    //setup clocks and hardware

```

```

EasyNucleoInit();
spiInit(); // Initialize SPI pins and clocks
GPIOInit(); // Initialize GPIO Pins

//Setup the LIS3DH for use
spiWrite(0x20, 0x77); // highest conversion rate, all axis on
spiWrite(0x23, 0x88); // block update, and high resolution

// Check WHO_AM_I register. should return 0x33
debug = spiRead(0x0F);

while(1)
{
    // Collect the X and Y values from the LIS3DH
    // The offsets were determined by Ho and Harris
    // They should really be checked for each LIS3DH
    // OFFSET: no tilt: 112; neg tilt: -15632; pos tilt: 12800
    x = spiRead(0x28) | (spiRead(0x29) << 8);
    // OFFSET: no tilt: 2224; neg tilt: -15872; pos tilt: 15840
    y = spiRead(0x2A) | (spiRead(0x2B) << 8);

    // correct for offset, scale appropriately
    disx = clip((272-x)/1000, 3) + 3;
    disy = clip((2000-y)/1000 - 1.5, 3) + 3; //The floating point
    // 1.5 slows down processing slightly, but not enough to notice
    // in the display. 1 was too little and 2 was too much to get a
    // stable dot in the center. Hence 1.5 (res 2017-11-09)

    // Display corresponding dot on matrix
    driveLED(disx,disy);

    // Small delay to reduce flicker
    delayLoop(100);
}
}

```

4. [3] Does your program work? Can you tilt the accelerometer and move the LED bubble across the array in both axes as expected and reach every dot in the 7x7 array?

YES.

If you have suggestions for further improvements of this lab, you're welcome to include them at the end of your lab.