1 Introduction

In this lab, you will use the ATMega328P's SPI controller to interface with an LIS3DH accelerometer. You will also implement software-driven PWM to control LEDs, using their intensity to indicate the force reading along each axis of the accelerometer.

Part	Due Date
Part A	Feb. 28
Part B	Mar. 6
Part C	Mar. 13
Reflection	Mar. 13

Figure 1: Table of due dates for each part.

Contents

1 Introduction **1**

2 Parts List **1**

3 Part A: LIS3DH Data Readout **2**
3.1 Wiring . . . 2
3.2 Requirements . . . 4
3.3 Grading . . . 4
3.4 Commentary . . . 4

4 Part B: LED Indicators **4**
4.1 Wiring . . . 4
4.2 Requirements . . . 6
4.3 Grading . . . 6
4.4 Commentary . . . 6

5 Part C: Combining the LIS3DH and LEDs **6**
5.1 Wiring . . . 6
5.2 Requirements . . . 7
5.3 Commentary . . . 7

6 Reflection **7**
6.1 Question 1 . . . 7
6.2 Question 2 . . . 7
6.3 Question 3 . . . 7
6.4 Question 4 . . . 7

7 Rubric **7**

2 Parts List

- one Adafruit Metro 328[1]
- six LEDs
- several male-male jumper wires
- several male-female jumper wiers
- one USB cable
- one breadboard
- one LIS3DH breakout board
- six 330Ω resistor (orange, orange, brown, gold)[2]
- AVRIce Programmer and cables

[1] Or compatible development board with an ATMega328P.
[2] Larger resistor may be substituted if needed.

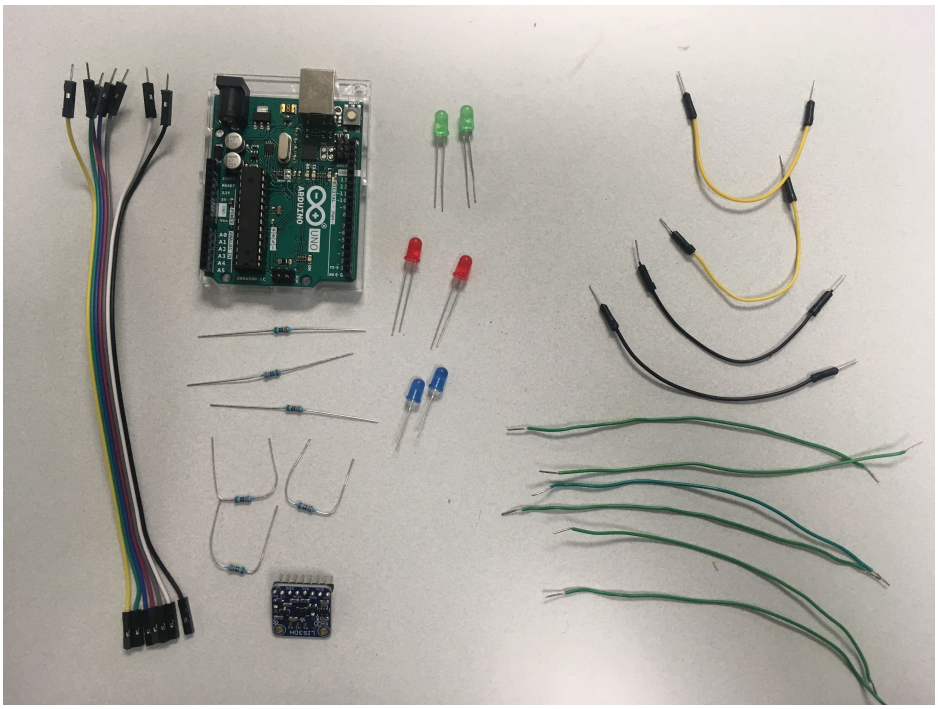


Figure 2: Photo showing parts for this project. Programmer and cables are not pictured, but are also needed.

3 Part A: LIS3DH Data Readout

3.1 Wiring

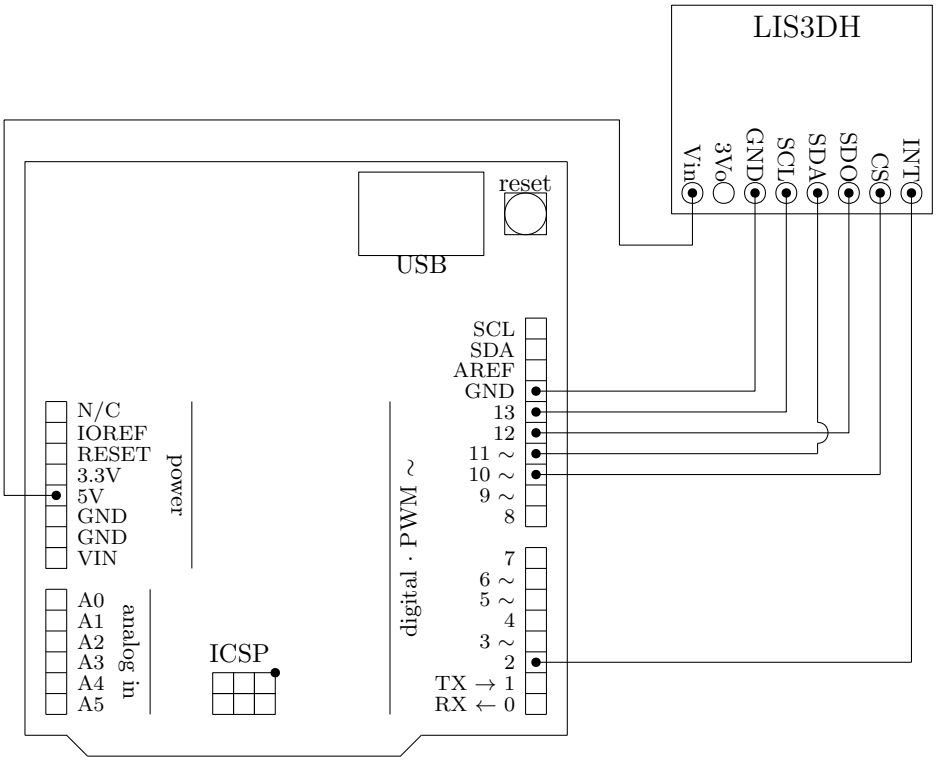


Figure 3: Wiring diagram for part A.

LIS3DH Pin	Metro 328 Pin	ATMega328P Pin
Vin	5V	5V
3Vo	N/C	N/C
GND	GND	GND
SCL	13	PB5/SCK/PCINT5
SDA	~ 11	PB3/MOSI/OC2A/PCINT3
SDO	12	PB4/MISO/PCINT4
CS	~ 10	PB2/SS/OC1B/PINT2
INT	2	PD2/INT0/PCINT18

Figure 4: Table showing correct connections for Part A.

You should also connect the USB cable from your Arduino to your workstation. This is how you will connect to the UART console.

Finally, connect the AVRIce programmer to your workstation, and the programming cable to the Arduino. Remember that the tab on the programming cable should be facing the interior of the Arduino board.

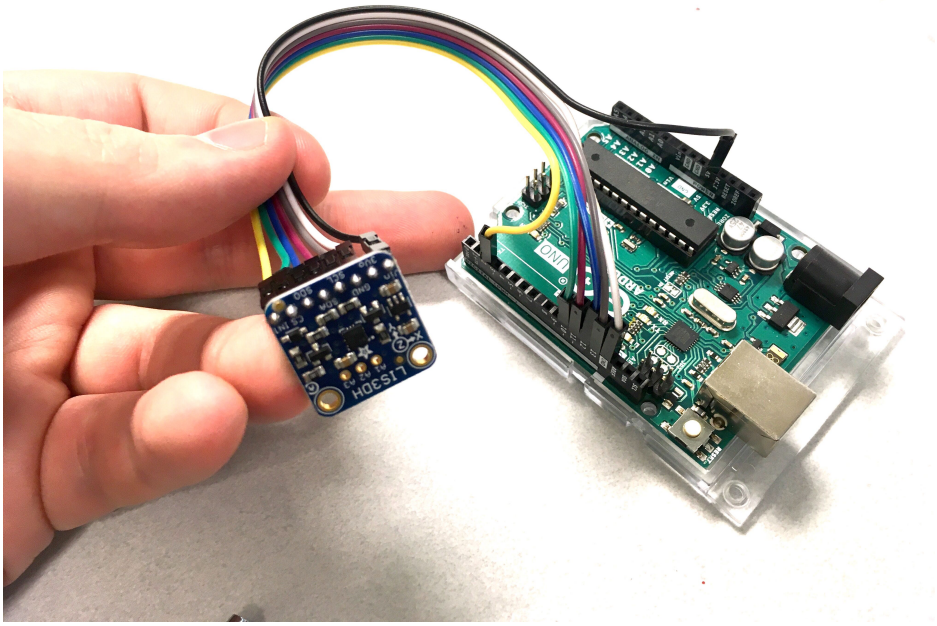


Figure 5: Photograph showing correctly wired project for part A (programmer and cables not pictured).

If you want to validate your wiring, you can use `make wiringtest` which will flash a test program to the board. Below are shown the expected outputs if the LIS3DH is wired correctly, or if it is wired incorrectly.

```
UART initialized (57600 8N1)


SPI initialized
initialization complete.


WHO_AM_I register read successfully!
Initial register values:
CTRL_REG0: 0x07
CTRL_REG1: 0x00
CTRL_REG2: 0x00
```



```
CTRL_REG3: 0x00
CTRL_REG4: 0x00
CTRL_REG5: 0x00
```

Listing 1: `make wiringtest` output with **correct** wiring.

```
UART initialized (57600 8N1)


SPI initialized
initialization complete.


WHO_AM_I register read failed!

Expected 0x33, got 0x0
Initial register values:
CTRL_REG0: 0x00
CTRL_REG1: 0x00
CTRL_REG2: 0x00
CTRL_REG3: 0x00
CTRL_REG4: 0x00
CTRL_REG5: 0x00
```

Listing 2: `make wiringtest` output with **incorrect** wiring.

3.2 Requirements

- (A.1) Your program should interface with the LIS3DH using SPI, and display the raw readings on each axis once per second over UART.

3.3 Grading

You do not need to submit any code for this portion. During lab, office hours, or by appointment, demonstrate your project to your TA or instructor.

3.4 Commentary

To avoid wasting many cycles polling, you will want to configure the LIS3DH to provide external interrupts to the ATMega. Once things are configured correctly, the code should be fairly simple.

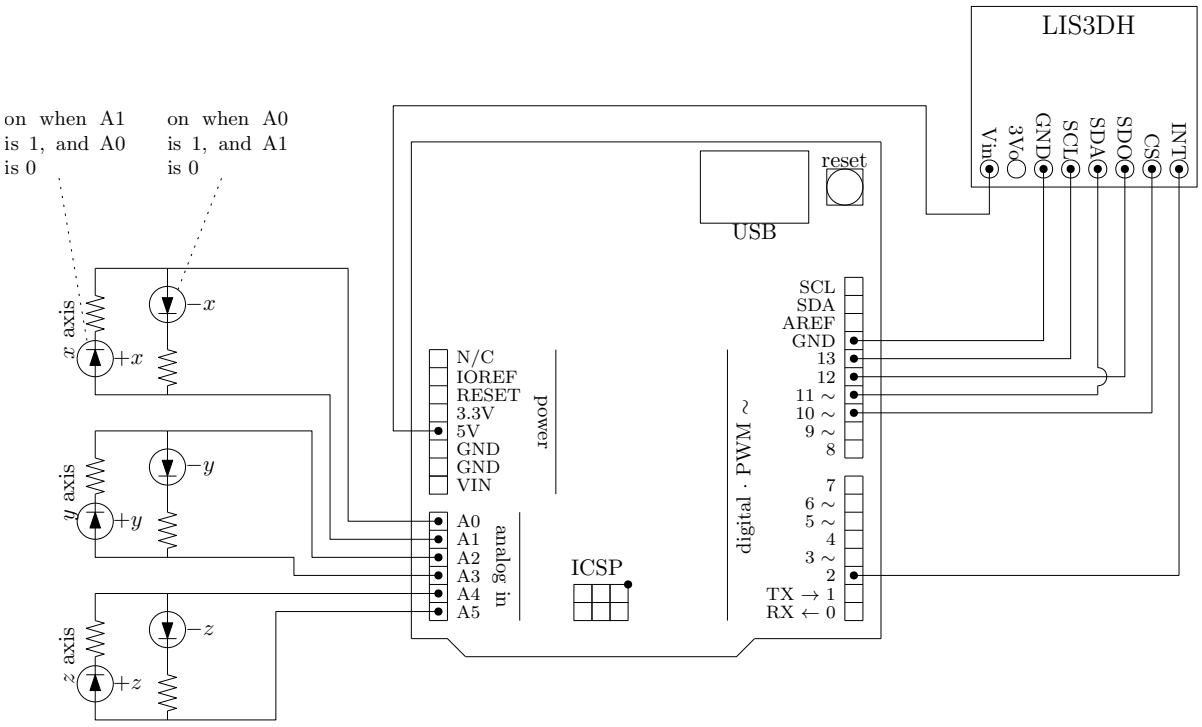
You will need to set `CTRL_REG5` to disable the FIFO, and latch interrupts. You will then want to set the `FIFO_CTRL_REG` to run in Bypass mode. Then you will want to set `CTRL_REG3` to enable the `DRDY1` interrupt. Lastly, set `CTRL_REG6` to allow the INT1 interrupt on the LIS3DH to be active low.

If you correctly wrangled all of those settings, you can then configure the ATMega to detect external active low interrupts, and in the ISR, you will first read the `STATUS_REG` register (to acknowledge/reset the interrupt), and then you can read the 6 output registers as normal.

4 Part B: LED Indicators

4.1 Wiring

For part B, you will connect six LEDs, each of which will be used to signify the intensity of the accelerated reading along the marked axis. Note that you will not be using the LIS3DH in this part, so you may optionally disconnect it.



All resistor values should be $\geq 330\Omega$

Figure 6: Wiring diagram for part B.

You will need to connect six LEDs. Each pair of pins in A0 ... A5, is used to drive one pair of LEDs - since the GPIO pins can be configured as positive or negative (pull up or pull down), thus allowing either LED in a pair to be activated at any given time.

You can test if your wiring is correct using the command `make blinkenlights`. This will allow you to toggle each LED on or off individually using the UART console (all LEDs will be off until you turn one on).

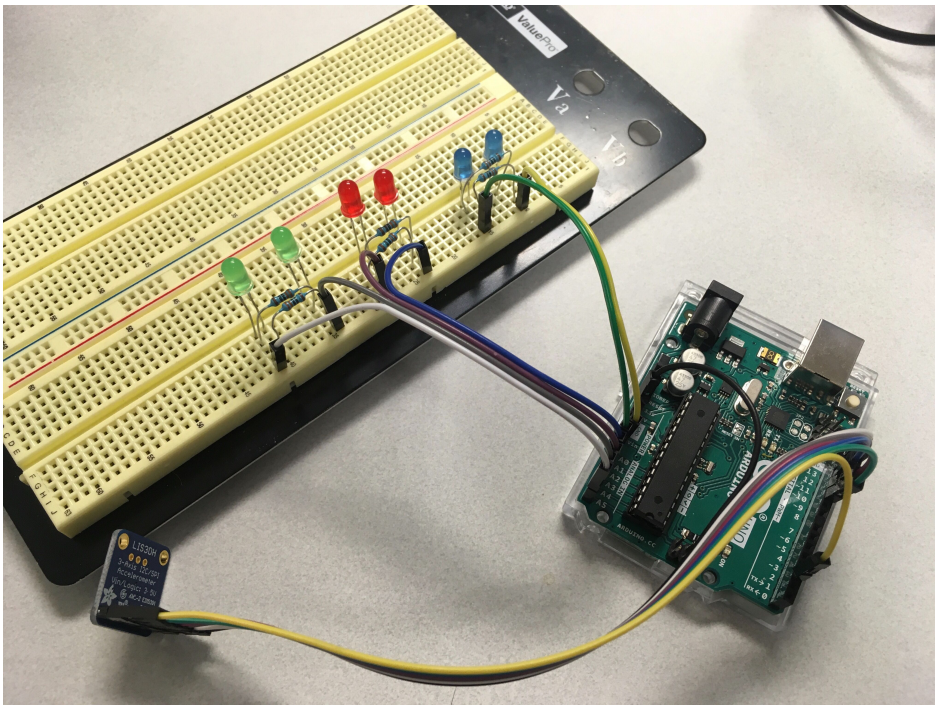


Figure 7: Photograph showing correctly wired project for part B (programmer and cables not pictured).

4.2 Requirements

- (B.1) Your program should expose an interactive command-line over UART. You should be able to enter three signed floating point values in the range $0.0 \ldots 1.0$, in the format `"%f %f %f"`. These values should represent "fake" sensor data for the x, y, and z axes respectively. You may implement other commands for your own debugging purposes as desired.
- (B.2) LEDs should take on a level of brightness proportional to the specified axis values. For example, if the user inputs `1.0 0.3 -0.7`, then the $+x$ LED should run at full brightness, the $+y$ LED should run at 30% brightness, and the $-z$ LED should run at 70% brightness. The $-x$, $-y$, and $+z$ LEDs should all be completely off.

4.3 Grading

You do not need to submit any code for this portion. During lab, office hours, or by appointment, demonstrate your project to your TA or instructor.

4.4 Commentary

This part segues into part C, in that you will simply be modifying your code to compute the duty cycles for each LED from data read from the LIS3DH, rather than the console.

As the ATMega328P has only two hardware PWM controllers, you will need to implement software PWM. To avoid floating point math, it is suggested that you should internally represent the PWM duty cycle as a `uint8_t`.

5 Part C: Combining the LIS3DH and LEDs

5.1 Wiring

The wiring for this part is unmodified from part B, except that LIS3DH sensor should be connected to the board if you disconnected it for part B.

5.2 Requirements

- (C.1) When the LIS3DH sensor is rotated, the LED brightness should change to show the intensity of the sensor reading. For example, when the LIS3DH is accelerated along it's $+x$ axis, the $+x$ LED should be illuminated brightly.
- (C.2) The LEDs should not flicker, and should have no perceptible delay before updating. The interrupt rate of the LIS3DH should be set to at least 25Hz.

5.3 Commentary

Although it was possible in part A to interface with the LIS3DH using either a polling or interrupt driven approach, part C will be much easier to complete successfully with an interrupt-driven method.

6 Reflection

You should submit your answers by editing `reflection.txt` to include them before you run `make pack`. There is no specific requirement for word or page count, but roughly 1-3 paragraphs are expected in response to each question. Note that your reflection will be graded with your submission to part D.

6.1 Question 1

For what reason might the `WHO_AM_I` register be included in the design of the LIS3DH? What negative impacts might there be if it were omitted?

6.2 Question 2

What value should be written to the LIS3DH's `CTRL_REG1` in order to achieve an data rate of 200Hz in normal mode? How do you know? Why do you think this value is implemented as a look-up table rather than as a pre-scaler, or written directly (i.e. why look up a 4 bit value in a table, rather than simply write 200.0, or a pre-scaling coefficient)?

6.3 Question 3

Why are we using an external interrupt instead of an SPI receive interrupt for this assignment? (Hint: Remember that the ATMega is configured as an *SPI master*.)

6.4 Question 4

Would it be possible to implement a software-based SPI slave by detecting a falling-edge triggered external interrupt?

7 Rubric

- 10 pts - Part A demonstration
- 10 pts - Part B demonstration
- 10 pts - Part C demonstration
- 20 pts - Part C code review
  - 5 pts - Use of SPI controller.
  - 5 pts - Correct interfacing with LIS3DH.
  - 5 pts - Correct interrupt-handling with the LIS3DH.
  - 5 pts - Software PWM implementation.
  - 5 pts - Style.
- 30 pts - Reflection (TBD pts per question)

Maximum number of points possible: 85.

Keep in mind that some items not listed on the rubric may cause you to loose points, including cheating, submitting code which is inconsistent with what you have demonstrating, plagiarizing code or reflection content, etc.