ARL-MR-1045 ● Nov 2021

DEVCOM ARMY RESEARCH LABORATORY

Designing a 3D Printable Arduino Integrated Circuit for Acceleration Sensing on Smart Munitions: Part I. Breadboarding

by Nikhil Murty, Harvey Tsang, and Jian Yu

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-MR-1045 ● Nov 2021

Designing a 3D Printable Arduino Integrated Circuit for Acceleration Sensing on Smart Munitions: Part I. Breadboarding

Harvey Tsang
Sensors and Electron Devices Directorate, DEVCOM Army Research Laboratory

Jian Yu
Weapons and Materials Research Directorate,
DEVCOM Army Research Laboratory

Nikhil Murty
Oak Ridge Associated Universities (ORAU)

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
November 2021	Memorandum Report	1 October 2020 to 30 September 2021

4. TITLE AND SUBTITLE
Designing a 3D Printable Arduino Integrated Circuit for Acceleration Sensing on Smart Munitions: Part I. Breadboarding

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Nikhil Murty, Harvey Tsang, and Jian Yu

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
DEVCOM Army Research Laboratory
ATTN: FCDD-RLW-MD
Aberdeen Proving Ground, MD 21005

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-MR-1045

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES
ORCID ID: Jian Yu, 0000-0002-1681-4602

14. ABSTRACT
In supporting the Army's modernization priority, the US Army Combat Capabilities Development Command Army Research Laboratory is designing new materials and processes to manufacture smart munitions with enhanced precision and lethality, while optimizing size, weight, power and cost. One manufacturing technology we are developing is additive manufacturing or 3D printing of electronics. This technical report is Part I of a workflow process to design a 3D printable Arduino integrated circuit, which will be used in a smart 40-mm grenade dome to sense acceleration. The breadboarding routine is presented here for a demonstrator circuit and its functionality is demonstrated by sensing acceleration.

15. SUBJECT TERMS
additive manufacturing, 3D printing, electronics, smart munitions, breadboarding

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Jian Yu
Unclassified	Unclassified	Unclassified	UU	23	19b. TELEPHONE NUMBER (Include area code) (410) 306-0698

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

1. Introduction

"The Army must modernize today or we could lose the next war," wrote the former Secretary of the Army Ryan D McCarthy.[1] One of the modernization priorities set forward by the US Army Futures Command is Long-Range Precision Fires munitions. In supporting this priority, the US Army Combat Capabilities Development Command Army Research Laboratory is designing new materials and processes to manufacture smart munitions with enhanced precision and lethality, while optimizing size, weight, power and cost (SWaP-C). One manufacturing technology that we are developing is printed hybrid electronics or hybrid printed electronics, also known as additive manufacture or 3D printing of electronics. The integrated circuits are printed as part of the 3D printing production process. All the electronics are conformal or embedded directly into structural components, where SWaP-C are minimized. To incorporate 3D integrated circuit onto a structure, we convert traditional integrated circuitry into a 3D circuit. Here, we present a workflow process where we design a 3D printable Arduino integrated circuit for acceleration sensing in a 40-mm grenade dome.

2. Circuit

2.1 Breadboard Setup

A breadboard circuit is developed to test the efficacy of the circuit before printing it. The main components include an accelerometer (ADXL337, made by Sparkfun), six different colored LEDs, an ATMEGA329-PU microcontroller, an Arduino Uno, and a USBasp programmer. The accelerometer reads the trajectory of the mortar up to 3 G's but it can be scaled up to 200 G's if using the ADXL377. The LEDs light up if the device records a positive or negative acceleration in the X, Y, and Z planes. The microcontroller controls the entire circuit and converts the data coming from the accelerometer from a voltage (0–3.3 V) to acceleration and then controls which LEDs turn on. The USBasp programmer is the interface between the circuit and the computer so that the microcontroller can be programmed. The Arduino Uno is used to supply 3.3 V to the accelerometer in lieu of a voltage regulator that will be used on the printed board.

As shown in Fig. 1, the analog input pins that are used for the accelerometer are in the top right corner (pins 23 to 28). The Master Out Servant In (MOSI), Master In Servant Out (MISO), and Serial Clock (SCK) pins are on the bottom right (pins 17 to 19) and these are connected to the USBasp programmer. The ground, Voltage Common Collector, and reset pins are also connected to the USBasp. The pins

labeled PD and PB (port names) are the input/output pins that can be connected to the LED.

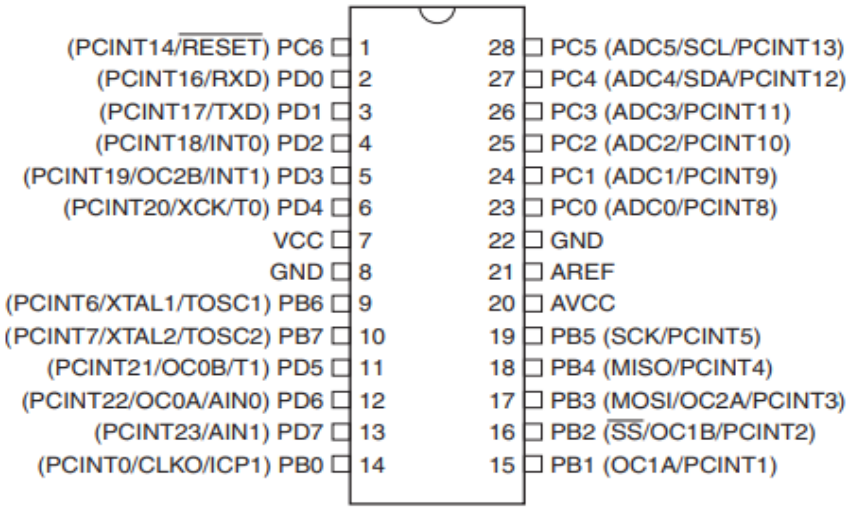


Fig. 1 Diagram of the pinout for the ATMEGA328-PU[2]

Figure 2 shows the setup of the circuit where green wires denote ground, red wires indicate 5 V, and white wires are 3.3 V. The blue wires connect the accelerometer (red square) to the analog input pins 23 to 25. When looking at the USBasp device (navy blue), black and orange wires are the MOSI, MISO, and SCK connections and the purple wire is the reset wire. A 10K-Ω resistor connects the reset pin on the microcontroller to 5-V power. The pink wires connect the anodes of the LEDs to the microcontroller, while the cathodes are connected to 220-Ω resistors, which are connected to ground. One of the LEDs does use the MOSI pin; however, this does not cause any interference. Figure 3 shows the completed circuit; however, the wire colors are different but the layout is the same.

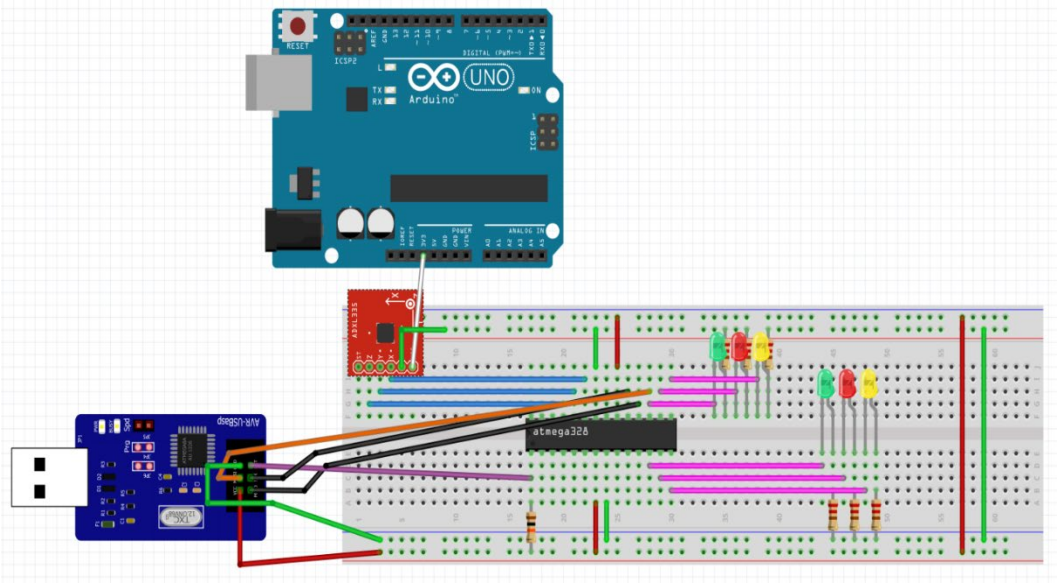


Fig. 2 Wiring layout

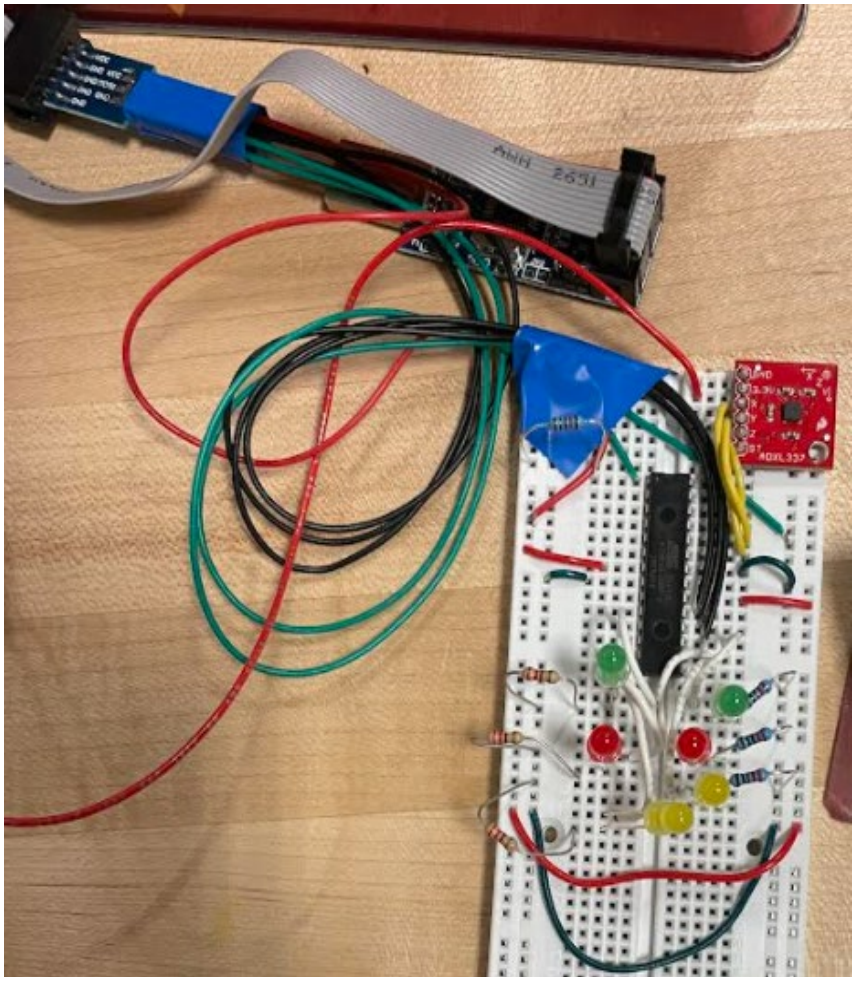
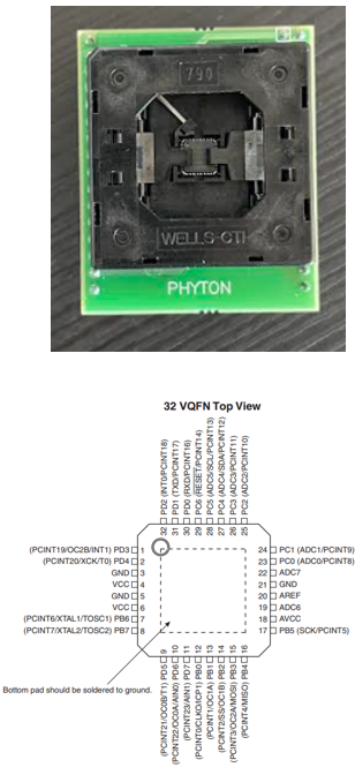


Fig. 3 Breadboard setup

2.2 The Quad Flat No-Leads (QFN) Setup

A circuit is made to program and test the QFN versions of the ATMEGA328 before it is added into the mortar shell. The QFN chip is placed into a dual in-line package (DIP)32 adapter that allows the QFN chip to be put onto a breadboard. However, it has two extra pins that are not used. Figure 4 shows the pinout for the QFN chip as well as the DIP32 adapter used. Following this new pinout of the QFN setup, the wires need to be rerouted to connect to the board, as shown in Fig. 5.

Pin (Arduino Pin)	DIP32
E-Pin	1
1 (3)	2
2 (4)	3
3 (GND)	4
4 (VCC)	5
5 (GND)	6
6 (VCC)	7
7 (20)	8
8 (21)	9
9 (5)	10
10 (6)	11
11 (7)	12
12 (8)	13
13 (9)	14
14 (10)	15
15 (11 – MOSI)	16
16 (12 – MISO)	17



Pin (Arduino Pin)	DIP32
E-Pin	34
32 (2)	33
31 (1)	32
30 (0)	31
29 (22-Reset)	30
28 (A5/19)	29
27(A4/18)	28
26 (A3/17)	27
25 (A2/16)	26
24 (A0/15)	25
23 (A0/14)	24
22 (A7)	23
21 (GND)	22
20 (AREF)	21
19 (A6)	20
18 (AVCC)	19
17 (13 –SCK)	18

Fig. 4 QFN and DIP32 pinout

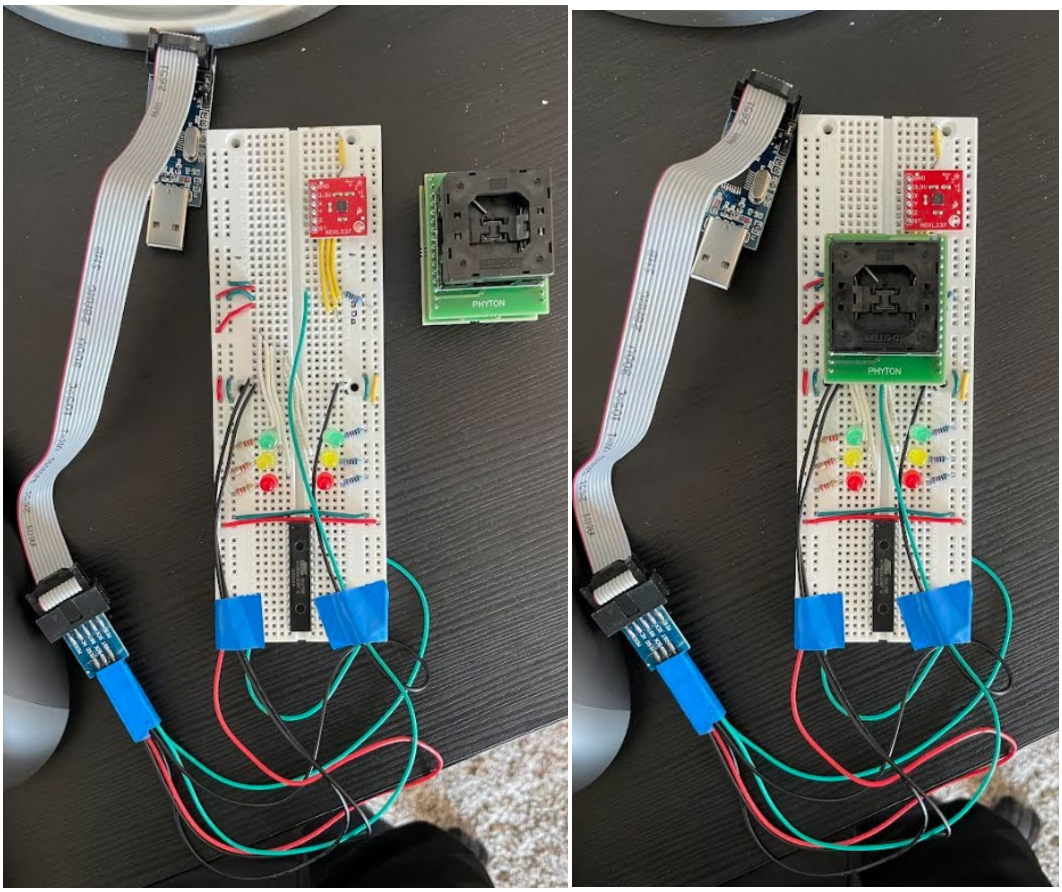


Fig. 5 Circuit with the QFN board

3. The Program

3.1 Software

First, the computer needs to be setup to recognize the USBasp device by following the instructions to install the necessary drivers: https://www.instructables.com/USBASP-Installation-in-Windows-10/.

Next, the Arduino Integrated Development Environment (IDE) needs to be configured to allow the microcontroller to use the internal 8-MHz clock instead of adding an external clock. This can be done by following the instructions from here: https://www.instructables.com/Configure-Arduino-IDE-for-Atmega-328P-to-Use-8MHz-/. After this has been set up, go in to the tools section of the IDE again and change the programmer to USBasp. To upload any code to the microcontroller, the user has to go in to the sketch section and click Upload Using Programmer and the USBasp device will upload the Arduino sketch onto the microcontroller.

3.2 Programming the Microcontroller

The code works by reading the accelerometer and then turns on the LEDs based on which direction the device is accelerating toward. For example, if the device is moving 1 G in the positive X direction, the positive X LED will turn on. Also, if there is no movement at all in a certain plane, then neither of the LEDs corresponding to that plane will turn on.

No specific libraries are needed to read the accelerometer, but some definitions need to be made at the start. A scale is required to see the maximum range of the accelerometer (3 G's if using an ADXL337 or 200 G's if using an ADXL377). An optional step is to define if a 5-V microcontroller is used, but this is not necessary and the code can be edited if only a 5-V controller will be used. The pins for the LEDs are also defined at the start. In the setup, the LED pins are initialized to output, and then all the LEDs flash to signal that the circuit is on. In the loop, the code starts by reading the values from the accelerometer. However, these values are voltages that have been converted into values ranging from 0 to 675 since we used a 5-V microcontroller. A map function then converts these values into G's, which are then scaled to the range of the accelerometer. Afterward, the LEDs turn on for each of the planes based on if the scaled values are greater or less than 0. The full code is included in the Appendix.

4. Results

Once the microcontroller has been programmed, the circuit is tested for the proper responses. Figure 6 shows the circuit when it is sitting flat on a table. In this case, all the LEDs should be off, but the accelerometer output is very rarely zero. A threshold could be added to correct this, but it was left on while testing the circuit. In general, the right-side LEDs indicate when the circuit moves in a positive direction and the left side indicates negative directions. The green LEDs are for the X axis, yellow is for the Y axis, and red is for the Z axis.

Figures 7, 8, and 9 show the circuit LEDs respond correctly when the board changes in the X, Y, and Z directions, respectively. The breadboard process designs the route for the circuit, which will then be printed on the 40-mm grenade dome

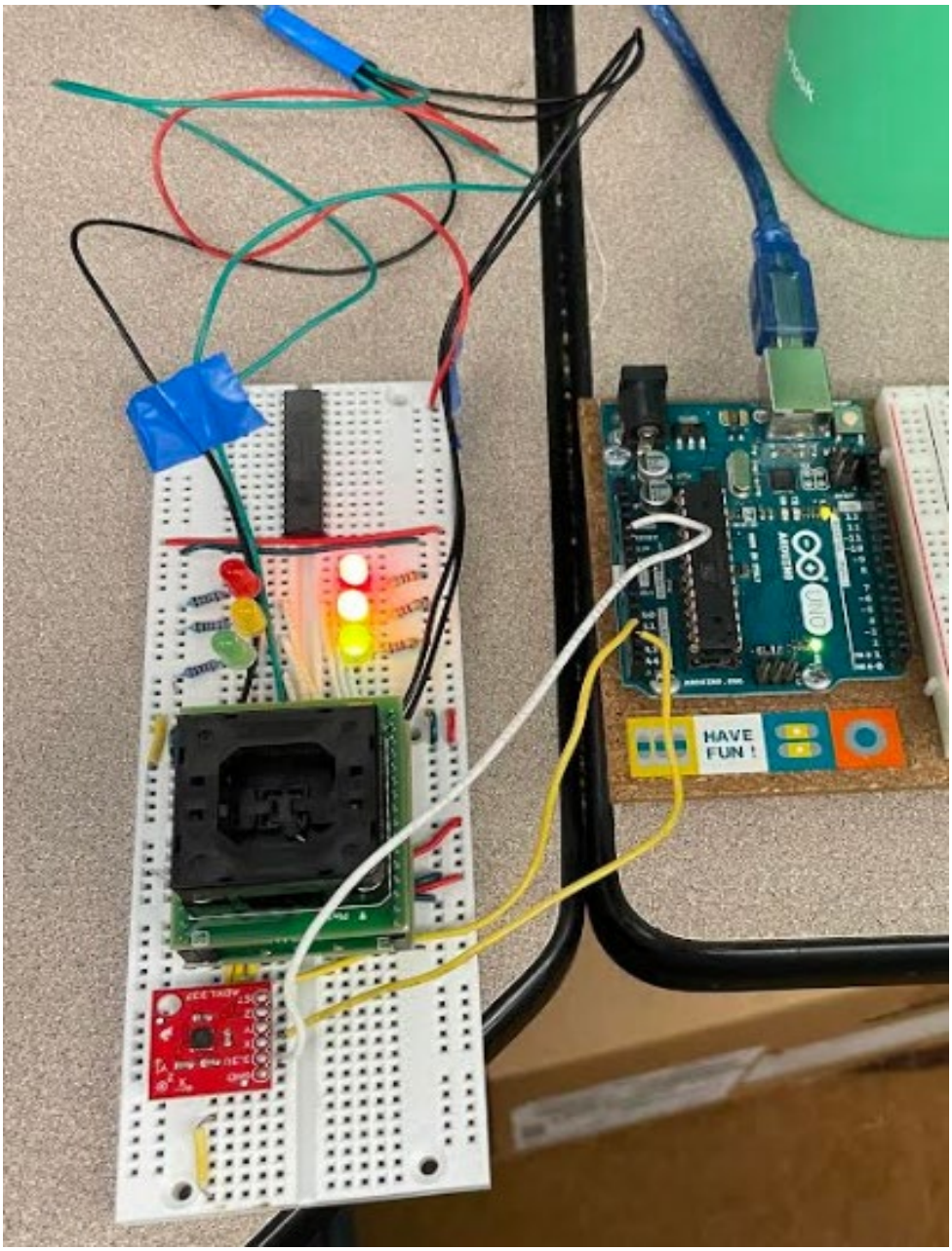


Fig. 6 Circuit in use with the QFN board

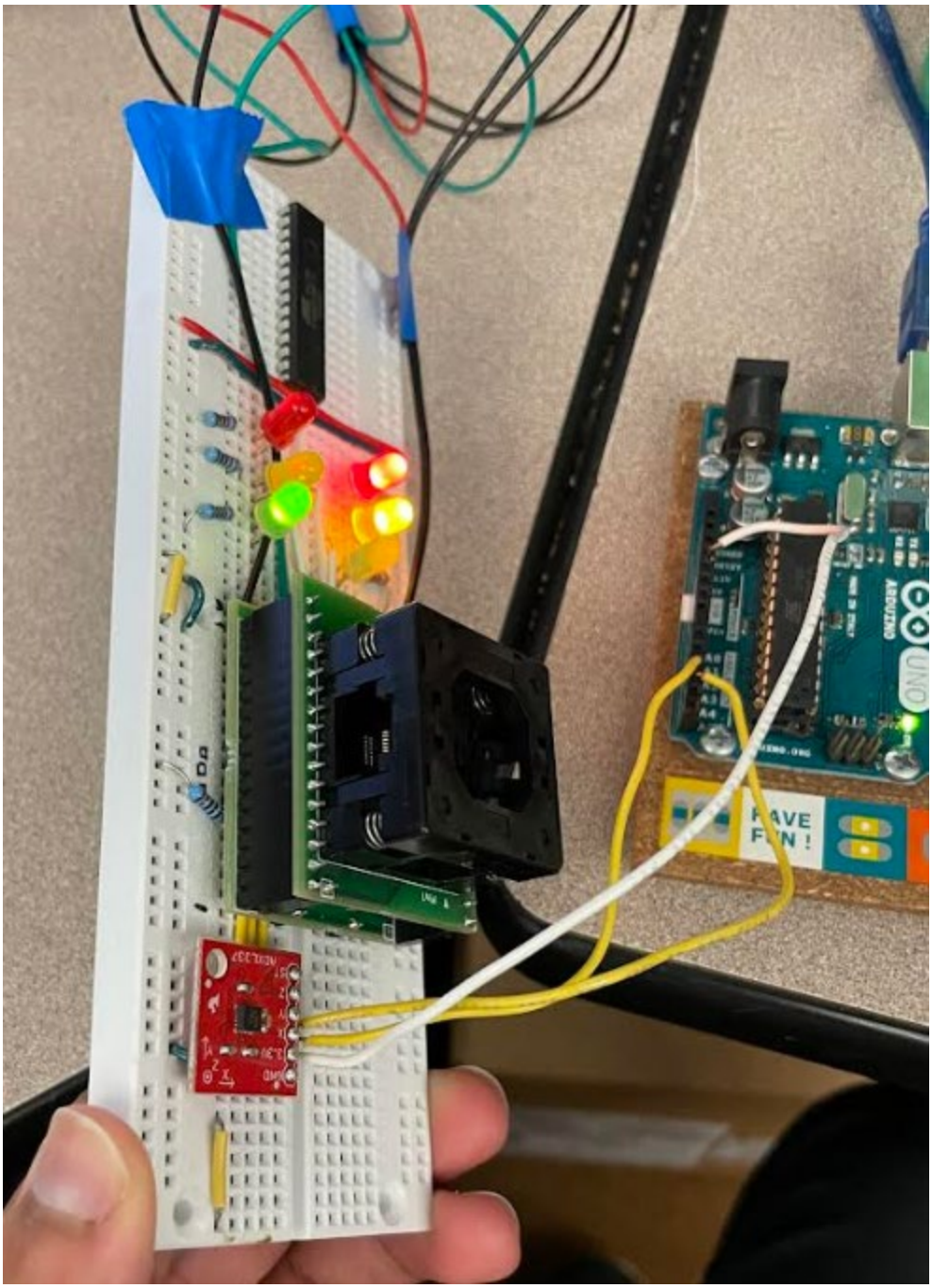


Fig. 7 Movement in the negative X direction

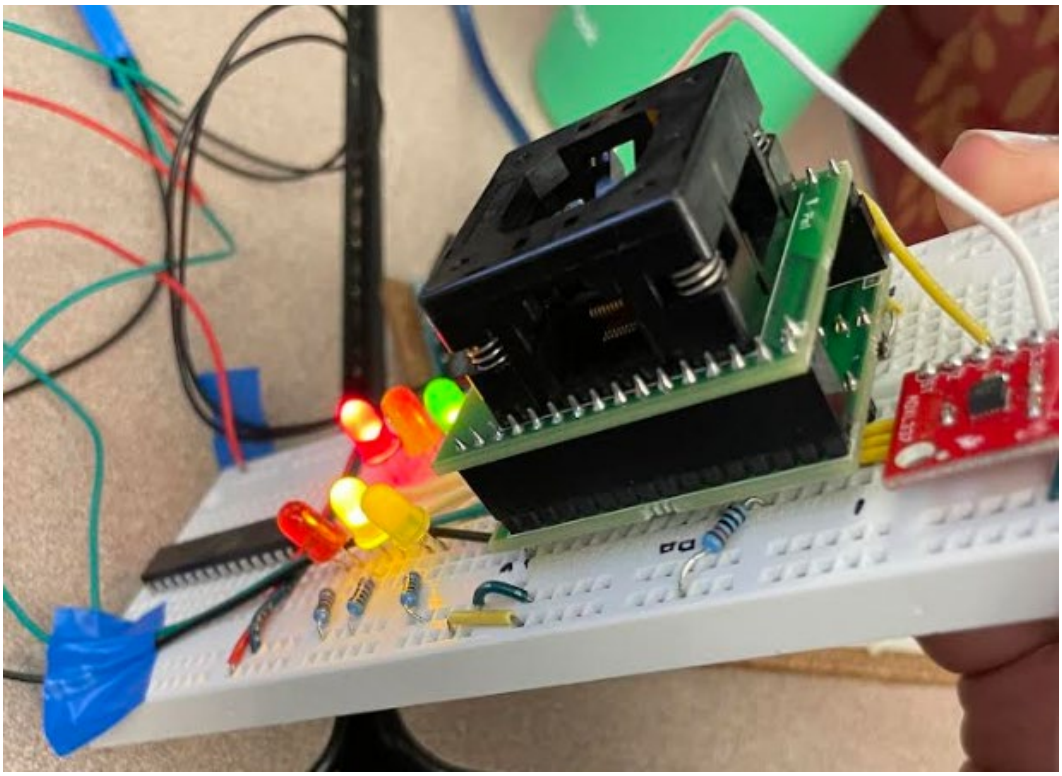


Fig. 8 Movement in the negative Y direction

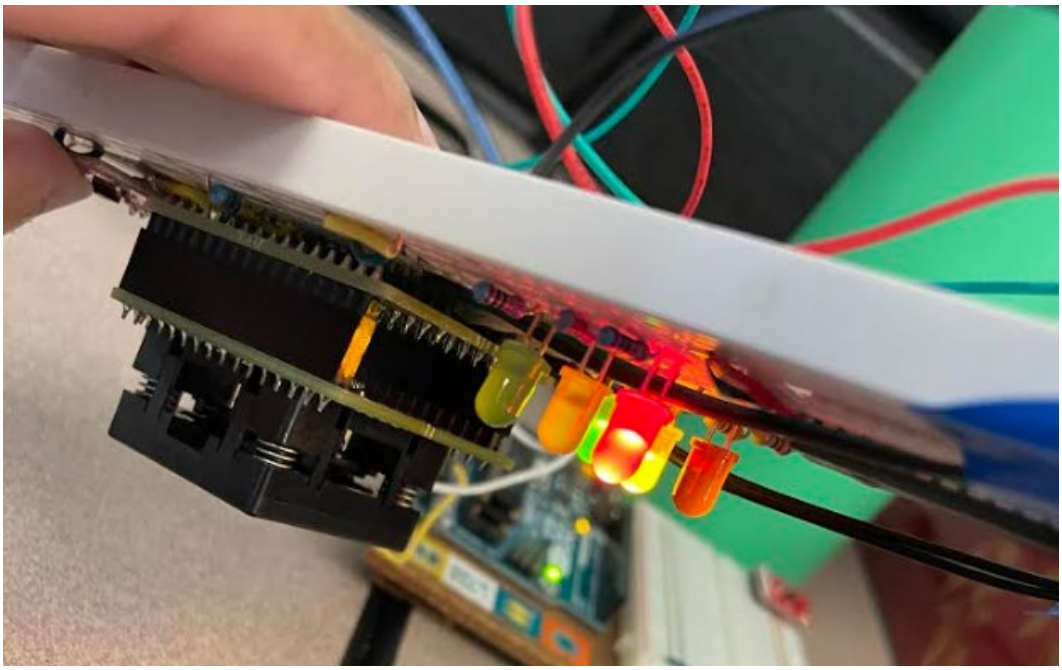


Fig. 9 Movement in the negative Z direction

5. Future Work

Part II in this series will discuss the next step, which is to begin making the printed circuits for the smart 40-mm grenade dome.

6. References

1. Fountain PB. Army Futures Command to highlight modernization efforts at 2019 AUSA. US Army Futures Command; 2019 October 9. https://www.army.mil/article/228297

2. ATMEGA328-PU data sheet. Atmel Corporation; 2012.

Appendix. Microcontroller Code

```
// Make sure these two variables are correct for your setup
int scale = 3; // 3 (±3g) for ADXL337, 200 (±200g) for ADXL377
boolean micro_is_5V = true; // set to true if using a 5V microcontroller

// LED setup
#define x_pos 6
#define x_neg 11
#define y_pos 7
#define y_neg 10
#define z_pos 8
#define z_neg 9

void setup()
{
  // Initialize serial communication at 9600 baud
  Serial.begin(9600);

  pinMode(x_pos, OUTPUT);
  pinMode(x_neg, OUTPUT);
  pinMode(y_pos, OUTPUT);
  pinMode(y_neg, OUTPUT);
  pinMode(z_pos, OUTPUT);
  pinMode(z_neg, OUTPUT);

  //startup
  digitalWrite(x_pos, HIGH);
  digitalWrite(x_neg, HIGH);
  digitalWrite(y_pos, HIGH);
  digitalWrite(y_neg, HIGH);
  digitalWrite(z_pos, HIGH);
  digitalWrite(z_neg, HIGH);
  delay(1000);
  digitalWrite(x_pos, LOW);
  digitalWrite(x_neg, LOW);
  digitalWrite(y_pos, LOW);
  digitalWrite(y_neg, LOW);
  digitalWrite(z_pos, LOW);
  digitalWrite(z_neg, LOW);
}

// Read, scale, and print accelerometer data
void loop()
{
  // Get raw accelerometer data for each axis
  int rawX = analogRead(A0);
  int rawY = analogRead(A1);
```

```
int rawZ = analogRead(A2);

// Scale accelerometer ADC readings into common units
// Scale map depends on if using a 5V or 3.3V microcontroller
float scaledX, scaledY, scaledZ; // Scaled values for each axis
if (micro_is_5V) // Microcontroller runs off 5V
{
  scaledX = mapf(rawX, 0, 675, -scale, scale); // 3.3/5 * 1023 =~ 675
  scaledY = mapf(rawY, 0, 675, -scale, scale);
  scaledZ = mapf(rawZ, 0, 675, -scale, scale);
}
else // Microcontroller runs off 3.3V
{
  scaledX = mapf(rawX, 0, 1023, -scale, scale);
  scaledY = mapf(rawY, 0, 1023, -scale, scale);
  scaledZ = mapf(rawZ, 0, 1023, -scale, scale);
}

/*
// Print out raw X,Y,Z accelerometer readings
Serial.print("X: "); Serial.println(rawX);
Serial.print("Y: "); Serial.println(rawY);
Serial.print("Z: "); Serial.println(rawZ);
Serial.println();

// Print out scaled X,Y,Z accelerometer readings
Serial.print("X: "); Serial.print(scaledX); Serial.println(" g");
Serial.print("Y: "); Serial.print(scaledY); Serial.println(" g");
Serial.print("Z: "); Serial.print(scaledZ); Serial.println(" g");
Serial.println();
*/

//LED output
// changes in x
if (scaledX > 0){
  digitalWrite(x_pos, HIGH);
  digitalWrite(x_neg, LOW);
}
else if (scaledX < 0) {
  digitalWrite(x_neg, HIGH);
  digitalWrite(x_pos, LOW);
}

// changes in y
if (scaledY > 0){
  digitalWrite(y_pos, HIGH);
```

```
  digitalWrite(y_neg, LOW);
 }
 else if (scaledY < 0) {
  digitalWrite(y_neg, HIGH);
  digitalWrite(y_pos, LOW);
 }
 // changes in z
 if (scaledZ > 0){
  digitalWrite(z_pos, HIGH);
  digitalWrite(z_neg, LOW);
 }
 else if (scaledZ < 0) {
  digitalWrite(z_neg, HIGH);
  digitalWrite(z_pos, LOW);
 }

 delay(10); // Minimum delay of 2 milliseconds between sensor reads (500 Hz)
}

// Same functionality as Arduino's standard map function, except using floats
float mapf(float x, float in_min, float in_max, float out_min, float out_max)
{
 return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```

List of Symbols, Abbreviations, and Acronyms

3D	three-dimensional
DIP	dual in-line package
IDE	Integrated Development Environment
LED	light-emitting diode
MISO	Master In Servant Out
MOSI	Master Out Servant In
QFN	quad flat no-leads
SCK	Serial Clock
USB	Universal Serial Bus

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD RLD DCI TECH LIB
3 (PDF)	DEVCOM ARL FCDD RLS EM H TSANG FCDD RLW MD J YU N MURTY