```
#include <Wire.h>
#include <Adafruit_LIS3DH.h>
#include <Adafruit_Sensor.h>


Adafruit_LIS3DH lis = Adafruit_LIS3DH();

// global constants:
const int ALPH = 26;
const int PIX = 5;
const int LEDS = 7;

// global variables:
  // variable to keep track of direction of motion:
  boolean forward;
  // array to keep track of mapping of output pins to LEDs:
  int ledNum[7] = {2,3,4,5,6,9,10};
  // variables used for average period calculation:
    int Tavg;
    int Tl;
    int Tl_prev;
    int Tl_cur;
    int Th;
    int Th_prev;
    int Th_cur;



void setup() {
  // initial waiting time:
  int initWaitTime = 200;

  //Serial.begin(9600);

  // start with left-to-right direction:
  forward = true;

  // initialize variables used for average period calculation:
  Tavg = 450;
  Tl = 0;
  Tl_prev = 0;
  Tl_cur = 0;
  Th = 0;
  Th_prev = 0;
```

```
  Th_cur = 0;

  if(!lis.begin(0x18))
  {
    Serial.println("Could not start!");
    while (1);
  }
  lis.setRange(LIS3DH_RANGE_2_G); //SET RESOLUTION (2,4,8,16)


  // configure output pins:
  for(int i=0; i<LEDS; i++) {
    pinMode(ledNum[i], OUTPUT);
  }

  // wait for initial waiting time:
  delay(initWaitTime);
}


void loop() {
  // word to display:
  String word1 = "TOOT";
  // length of word:
  int wordLength = word1.length();
  // total number of pixels:
  int pixTotal = PIX * wordLength;
  // total number of spaces:
  int spaces = wordLength - 1;
  // ratio of letterTime / pixTime:
  int ratio = 4;

  // time for whole word to display:
  int wordTime;
  // time in between pixels:
  int pixTime;
  // time in between letters:
  int letterTime;
  // "padding" time in between switching directions:
  int padTime = 200;
  // smaller "padding" time of while loop execution:
  int smallPadTime = 50;

  // accelerometer value to read in:
  int value;
```

```
  // acceleromter threshold values:
  int al = -13000;
  int ah = 13000;

  // letter encoding (7-bit binary):
  // only some letters were changed from a 5-bit encoding to a
7-bit encoding, because only these letters were used to display
words
  byte letters[ALPH][PIX] = {
                {3, 28, 100, 28, 3},    // A  (7 bits)
                {31, 21, 21, 10, 0},    // B  (5 bits)
                {14, 17, 17, 10, 0},    // C  (5 bits)
                {31, 17, 17, 14, 0},    // D  (5 bits)
                {127, 73, 73, 65, 0},   // E  (7 bits)
                {31, 20, 20, 16, 0},    // F  (5 bits)
                {14, 17, 19, 10, 0},    // G  (5 bits)
                {31, 4, 4, 4, 31},      // H  (5 bits)
                {0, 65, 127, 65, 0},    // I  (7 bits)
                {0, 17, 30, 16, 0},     // J  (5 bits)
                {31, 4, 10, 17, 0},     // K  (5 bits)
                {127, 1, 1, 1, 0},      // L  (7 bits)
                {31, 12, 3, 12, 31},    // M  (5 bits)
                {31, 12, 3, 31, 0},     // N  (5 bits)
                {14, 17, 17, 14, 0},    // O  (5 bits)
                {31, 20, 20, 8, 0},     // P  (5 bits)
                {14, 17, 19, 14, 2},    // Q  (5 bits)
                {31, 20, 22, 9, 0},     // R  (5 bits)
                {48, 73, 73, 6, 0},     // S  (7 bits)
                {64, 64, 127, 64, 64},  // T  (7 bits)
                {30, 1, 1, 30, 0},      // U  (5 bits)
                {24, 6, 1, 6, 24},      // V  (5 bits)
                {28, 3, 12, 3, 28},     // W  (5 bits)
                {17, 10, 4, 10, 17},    // X  (5 bits)
                {17, 10, 4, 8, 16},     // Y  (5 bits)
                {19, 21, 21, 25, 0}     // Z  (5 bits)
  };

  // update pixTime and letterTime:
  wordTime = .5*Tavg - padTime - smallPadTime;
  pixTime = (wordTime) / (pixTotal + (ratio*spaces));
  letterTime = ratio * pixTime;

  // display word (in one direction):
  displayWord(word1, wordLength, letters, forward, pixTime,
letterTime);
```

```
  // check to see when to switch direction while moving forward
(and calculate average period):
  bool forwardDisplayed = false;
  while(forward) {
    forwardDisplayed = true;

    // read in accelerometer:
    lis.read();
    value = lis.z;

    // check if value falls below low threshold value:
    if(value<al) {
      // switch direction:
      //Serial.println("Left to Right");
      forward = false;

      // record current time:
      Tl_cur = millis();
      //Serial.print("Tl_cur: ");
      //Serial.println(Tl_cur);
      //Serial.println();

      // calculate period:
      if(Tl_prev!=0) {
        Tl = Tl_cur - Tl_prev;
        //Serial.print("Tl: ");
        //Serial.println(Tl);
        //Serial.println();
      }
      // store value of Tl_cur:
      Tl_prev = Tl_cur;
    }
  }

  // check to see when to switch direction while moving backward
(and calculate average period):
  while(!forward && !forwardDisplayed) {
    // read in accelerometer:
    lis.read();
    value = lis.z;

    // check if value rises above high threshold value:
    if(value>ah) {
      // switch direction:
```

```
      //Serial.println("Right to Left");
      forward = true;

      // record current time:
      Th_cur = millis();
      //Serial.print("Th_cur: ");
      //Serial.println(Th_cur);
      //Serial.println();

      // calculate period:
      if(Th_prev!=0) {
        Th = Th_cur - Tl_prev;
        //Serial.print("Th: ");
        //Serial.println(Th);
        //Serial.println();
      }
      // store value of Th_cur;
      Th_prev = Th_cur;
    }
  }

  // at end of first swing, set Tavg to Tl:
  if(Tl!=0 && Th==0) {
    Tavg = Tl;
    //Serial.print("Tavg: ");
    //Serial.println(Tavg);
  }
  // calculate average period:
  if(Tl!=0 && Th!=0) {
    Tavg = (Tl+Th) / 2;
    //Serial.print("Tavg: ");
    //Serial.println(Tavg);
  }
  //Serial.println();

  // wait for "padding" time in between switching directions:
  delay(padTime);
}


// returns ascii code of input character
int getLetterIndex(char letter) {
  int ascii = '0' + letter;
  return ascii - 113;
}
```

```
// displays letter (forward or backward) on LEDs
void displayLetter(char letter, byte letters[ALPH][PIX], boolean
forward, int pixTime, int letterTime, int wordLength, int
letterNum) {
  int letterIndex = getLetterIndex(letter);
  Serial.println();
  Serial.println();
  Serial.println(letter);
  Serial.println();

  // display letter forwards:
  if(forward) {
    for(int pix=0; pix<PIX; pix++) {
      for(int i=0; i<LEDS; i++) {
        if(bitRead(letters[letterIndex][pix], i) == 1) {
          digitalWrite(ledNum[i], HIGH);
          //Serial.print('1');
        }
        else {
          digitalWrite(ledNum[i], LOW);
          //Serial.print('0');
        }
      }
      // wait in between pixels:
      delay(pixTime);
      //Serial.println();
    }
  }

  // display letter backwards:
  else {
    for(int pix=PIX-1; pix>=0; pix--) {
      for(int i=0; i<LEDS; i++) {
        if(bitRead(letters[letterIndex][pix], i) == 1) {
          digitalWrite(ledNum[i], HIGH);
          //Serial.print('1');
        }
        else {
          digitalWrite(ledNum[i], LOW);
          //Serial.print('0');
        }
      }
      // wait in between pixels:
```

```
      delay(pixTime);
      //Serial.println();
    }
  }

  // turn off all LEDs in between letters:
  for(int i=0; i<LEDS; i++) {
    digitalWrite(ledNum[i], LOW);
  }
  // wait in between letters (except for last letter displayed
in word):
  if((forward==true && letterNum!=wordLength-1) ||
(forward==false && letterNum!=0)) {
    delay(letterTime);
    //Serial.println();
  }
}


// displays word (forward or backward) on LEDs
void displayWord(String word1, int wordLength, byte
letters[ALPH][PIX], boolean forward, int pixTime, int
letterTime) {
  // display word forwards:
  if(forward) {
    for(int letter=0; letter<wordLength; letter++) {
      displayLetter(word1[letter], letters, forward, pixTime,
letterTime, wordLength, letter);
    }
  }

  // display word backwards:
  else {
    for(int letter=wordLength-1; letter>=0; letter--) {
      displayLetter(word1[letter], letters, forward, pixTime,
letterTime, wordLength, letter);
    }
  }
}
```