// this is my script that works great. Just limited to 120 words per sd card

/\*

  Catchphrase (No Categories)

  Hardware:

    - Arduino Uno

    - 1602 LCD (parallel) on pins RS=8, E=9, D4=A0(14), D5=A1(15), D6=A2(16), D7=A3(17)

    - Backlight control on A4 (18). 220 ohm resiter between this pin (A4) and LCD pin A

    - SD TF card reader (CS=10, MOSI=11, MISO=12, SCK=13)

    - piezo Buzzer on D7

    - Buttons:

        START/STOP = D2

        TEAM1      = D3

        TEAM2      = D4

        NEXT       = D5

        CATEGORY   = D6  (repurposed as MUTE toggle)

\*/

//text file (notepad) must be called words.txt

//12 characters per LCD line. The formatter tries to split on a space so the clue can use up to 2 lines.

//if a word is longer than 13 characters it gets skipped (no words longer than 13 characters)

// Blank lines or lines starting with # are ignored.

#include <SPI.h>

#include <SD.h>

#include <LiquidCrystal.h>

// ===== Pins (your mapping) =====

const byte TRANSISTOR\_POWER\_PIN = 19; // A5

const byte START\_STOP\_PIN = 2;

const byte TEAM1\_PIN      = 3;

const byte TEAM2\_PIN      = 4;

const byte NEXT\_PIN       = 5;  // Next word

const byte CATEGORY\_PIN   = 6;  // Mute toggle

const byte SPEAKER\_PIN    = 7;

const byte LCD\_PIN\_RS = 8;

const byte LCD\_PIN\_E  = 9;

const byte SD\_PIN\_CS  = 10;

const byte LCD\_PIN\_D4 = 14; // A0

const byte LCD\_PIN\_D5 = 15; // A1

const byte LCD\_PIN\_D6 = 16; // A2

const byte LCD\_PIN\_D7 = 17; // A3

const byte LCD\_PIN\_BL = 18; // A4

// ===== LCD + SD =====

LiquidCrystal lcd(LCD\_PIN\_RS, LCD\_PIN\_E, LCD\_PIN\_D4, LCD\_PIN\_D5, LCD\_PIN\_D6, LCD\_PIN\_D7);

File wordsFile;

// ===== Scores (global so showScoresAndText sees them) =====

int score\_team1 = 0;

int score\_team2 = 0;

// ===== Display formatting =====

// Top row: [scoreL][ 14-char text ][scoreR]  => total 16 columns

// Bottom row: full 16-char window

#define TOP\_TEXT\_LEN     14

#define BOTTOM\_TEXT\_LEN  16

String pad\_center(String text, uint8\_t width) {

  text.trim();

  if (text.length() > width) return "";   // too long for this line

  uint8\_t leftPad  = (width - text.length()) / 2;

  uint8\_t rightPad = width - text.length() - leftPad;

  String s;

  for (uint8\_t i=0;i<leftPad;i++)  s += ' ';

  s += text;

  for (uint8\_t i=0;i<rightPad;i++) s += ' ';

  return s;

}

// Format a clue to fit: TOP=14 chars, BOTTOM=16 chars.

// Returns a single String with length TOP\_TEXT\_LEN + BOTTOM\_TEXT\_LEN.

// First 14 chars -> top text window; last 16 chars -> bottom line.

// splits a phrase at a space if total phrase is longer than the top\_text length (14).

// if single word is longer than 14 letters, but no longer than 16, it is displayed on bottom line and top line is left empty (except the scores)

// any words 17+ letters is skipped.

String format\_for\_lcd(String text) {

  text.trim();

  auto pad\_center = [](const String &t, uint8\_t width) {

    String s = t; s.trim();

    if (s.length() > width) return String(""); // fail

    uint8\_t left  = (width - s.length()) / 2;

    uint8\_t right = width - s.length() - left;

    String out;

    for (uint8\_t i=0;i<left;i++)  out += ' ';

    out += s;

    for (uint8\_t i=0;i<right;i++) out += ' ';

    return out;

  };

  if (text.length() == 0) {

    return pad\_center("", TOP\_TEXT\_LEN) + pad\_center("", BOTTOM\_TEXT\_LEN);

  }

  // ---- NEW: single long token (no spaces) of 15–16 chars -> show on bottom line ----

  if (text.indexOf(' ') < 0 && text.length() > TOP\_TEXT\_LEN && text.length() <= BOTTOM\_TEXT\_LEN) {

    String top    = pad\_center("",   TOP\_TEXT\_LEN);     // blank top window

    String bottom = pad\_center(text, BOTTOM\_TEXT\_LEN);  // word centered on full bottom row

    return (bottom.length() == 0) ? String("") : (top + bottom);

  }

  // Fits entirely on top?

  if (text.length() <= TOP\_TEXT\_LEN) {

    String top    = pad\_center(text, TOP\_TEXT\_LEN);

    String bottom = pad\_center("",   BOTTOM\_TEXT\_LEN);

    return (top.length() == 0 || bottom.length() == 0) ? String("") : (top + bottom);

  }

  // Two-line split at last space that keeps top within 14

  int lastSpaceTop = text.lastIndexOf(' ', TOP\_TEXT\_LEN);

  if (lastSpaceTop < 0) {

    // First token exceeds 14 and has no spaces -> can't split (and >16 handled above)

    return String("");

  }

  String topPart = text.substring(0, lastSpaceTop);

  String botPart = text.substring(lastSpaceTop + 1);

  botPart.trim();

  if (botPart.length() > BOTTOM\_TEXT\_LEN) {

    return String(""); // bottom would overflow

  }

  String top    = pad\_center(topPart, TOP\_TEXT\_LEN);

  String bottom = pad\_center(botPart, BOTTOM\_TEXT\_LEN);

  return (top.length() == 0 || bottom.length() == 0) ? String("") : (top + bottom);

}

void lcdClearLine(byte row) {

  lcd.setCursor(0,row);

  for (byte i=0;i<16;i++) lcd.print(' ');

}

void showScoresAndText(const String &mainText) {

  // mainText comes from format\_for\_lcd(): first 14 chars top-window, last 16 bottom-line

  String topWin = mainText.substring(0, TOP\_TEXT\_LEN);

  String bot    = mainText.substring(TOP\_TEXT\_LEN); // 16 chars

  // Top: put left score at col0, right score at col15, and the 14-char window in between

  lcd.setCursor(0,0);  lcd.print(score\_team1);   // col 0

  lcd.setCursor(1,0);  lcd.print(topWin);        // cols 1..14

  lcd.setCursor(15,0); lcd.print(score\_team2);   // col 15

  // Bottom: draw the full 16-char centered line

  lcd.setCursor(0,1);

  lcd.print(bot);

}

// ===== Debounced buttons =====

struct DebouncedButton {

  byte pin;

  byte lastAdvertised;

  byte curAdvertised;

  byte lastRead;

  unsigned long lastChange;

  void begin(byte p) { pin=p; pinMode(p, INPUT\_PULLUP); lastAdvertised=curAdvertised=lastRead=HIGH; lastChange=0; }

  void update() {

    byte s = digitalRead(pin);

    unsigned long now = millis();

    if (s != lastRead) lastChange = now;

    if (now - lastChange > 50) curAdvertised = s;

    lastRead = s;

  }

  bool justPressed()  { bool jp = (curAdvertised != lastAdvertised) && (curAdvertised == LOW);  lastAdvertised = curAdvertised; return jp; }

  bool justReleased() { bool jr = (curAdvertised != lastAdvertised) && (curAdvertised == HIGH); lastAdvertised = curAdvertised; return jr; }

  bool isPressed()    { return curAdvertised == LOW; }

};

DebouncedButton btnStart, btnT1, btnT2, btnNext, btnMute;

// ===== Game state =====

enum GAME\_STATE { READY, IN\_ROUND, GAME\_DONE };

GAME\_STATE gameState = READY;

bool muted = false;

String currentWord;

// ===== Beep timing (speeds up) =====

unsigned long beep\_frequency\_change\_interval\_millis = 15000;

unsigned long beep\_interval\_millis[] = {500, 500, 300, 200};

const int NUM\_BEEP\_INTERVALS = 4;

int cur\_beep\_interval = 0;

bool next\_is\_tic = true;

unsigned long last\_tictoc\_millis = 0;

unsigned long last\_beep\_speed\_change\_millis = 0;

// ===== Random word index (offset table, shuffled in-place) =====

// Keep memory small for Uno

const uint16\_t MAX\_WORDS = 120;               // 120 words = ~480 bytes (120 \* 4B) — SAFE for Uno. 200 words = 800 bytes

unsigned long wordOffsets[MAX\_WORDS];

uint16\_t wordCount = 0;

uint16\_t wordPos = 0;

bool rng\_seeded = false;

void fisherYatesShuffleOffsets() {

  if (wordCount <= 1) return;

  for (int i = wordCount - 1; i > 0; --i) {

    int j = random(i + 1); // 0..i

    unsigned long tmp = wordOffsets[i];

    wordOffsets[i] = wordOffsets[j];

    wordOffsets[j] = tmp;

  }

}

bool indexWords() {

  wordCount = 0;

  wordsFile.seek(0);

  while (true) {

    unsigned long startPos = wordsFile.position();

    String line = wordsFile.readStringUntil('\n');

    if (line.length() == 0 && !wordsFile.available()) break; // EOF

    String copy = line; copy.trim();

    if (copy.length() == 0) continue;            // skip blanks

    if (copy.startsWith("#")) continue;          // skip comments

    if (format\_for\_lcd(copy).length() == 0) continue; // too long to display nicely

    if (wordCount < MAX\_WORDS) {

      wordOffsets[wordCount++] = startPos;

    } else {

      // hit cap — ignore remainder to save RAM

      break;

    }

  }

  wordPos = 0;

  fisherYatesShuffleOffsets();

  return wordCount > 0;

}

bool getNextRandomWord(String &out) {

  if (wordCount == 0) return false;

  if (wordPos >= wordCount) {

    fisherYatesShuffleOffsets();

    wordPos = 0;

  }

  unsigned long off = wordOffsets[wordPos++];

  wordsFile.seek(off);

  String line = wordsFile.readStringUntil('\n');

  line.trim();

  out = line;

  return out.length() > 0;

}

// ===== Beeps =====

void beep\_tic()      { if (!muted) tone(SPEAKER\_PIN, 300, 30); }

void beep\_toc()      { if (!muted) tone(SPEAKER\_PIN, 300, 30); }

void beep\_times\_up() {

  if (!muted) { tone(SPEAKER\_PIN, 300, 300); delay(300); tone(SPEAKER\_PIN, 300, 300); delay(300); tone(SPEAKER\_PIN, 300, 300); }

  else { delay(900); }

}

void beep\_power\_on() { if (!muted) tone(SPEAKER\_PIN, 300, 30); }

void beep\_small()    { if (!muted) tone(SPEAKER\_PIN, 300, 30); }

// ===== Round flow =====

void showWord(const String &word) {

  String two = format\_for\_lcd(word);

  if (two.length() == 0) two = format\_for\_lcd("(too long)");

  showScoresAndText(two);

}

void startRound() {

  gameState = IN\_ROUND;

  cur\_beep\_interval = 0;

  next\_is\_tic = true;

  last\_tictoc\_millis = 0;

  last\_beep\_speed\_change\_millis = millis();

  if (!getNextRandomWord(currentWord)) {

    lcd.clear(); lcd.setCursor(0,0); lcd.print(F("No words indexed"));

    lcd.setCursor(0,1); lcd.print(F("Check words.txt"));

    return;

  }

  showWord(currentWord);

}

void endRound(bool timesUp=true) {

  if (timesUp) beep\_times\_up();

  gameState = READY;

  String two = format\_for\_lcd("Press Start");

  showScoresAndText(two);

}

void do\_tic\_toc() {

  unsigned long now = millis();

  if (now - last\_beep\_speed\_change\_millis > beep\_frequency\_change\_interval\_millis) {

    last\_beep\_speed\_change\_millis = now;

    if (++cur\_beep\_interval >= NUM\_BEEP\_INTERVALS) { endRound(true); return; }

  }

  if (now - last\_tictoc\_millis > beep\_interval\_millis[cur\_beep\_interval]) {

    if (next\_is\_tic) beep\_tic(); else beep\_toc();

    next\_is\_tic = !next\_is\_tic;

    last\_tictoc\_millis = now;

  }

}

// ===== Setup / Loop =====

void setup() {

  pinMode(TRANSISTOR\_POWER\_PIN, OUTPUT);

  digitalWrite(TRANSISTOR\_POWER\_PIN, HIGH);

  pinMode(LCD\_PIN\_BL, OUTPUT);

  digitalWrite(LCD\_PIN\_BL, HIGH);

  pinMode(SPEAKER\_PIN, OUTPUT);

  btnStart.begin(START\_STOP\_PIN);

  btnT1.begin(TEAM1\_PIN);

  btnT2.begin(TEAM2\_PIN);

  btnNext.begin(NEXT\_PIN);

  btnMute.begin(CATEGORY\_PIN);

  lcd.begin(16,2);

  lcdClearLine(0); lcdClearLine(1);

  lcd.setCursor(0,0); lcd.print(F("Loading SD..."));

  pinMode(SD\_PIN\_CS, OUTPUT);

  digitalWrite(SD\_PIN\_CS, HIGH);

  if (!SD.begin(SD\_PIN\_CS)) {

    lcd.setCursor(0,1); lcd.print(F("SD FAIL"));

    while (1) { }

  }

  wordsFile = SD.open("words.txt", FILE\_READ);

  if (!wordsFile) wordsFile = SD.open("WORDS.TXT", FILE\_READ);

  if (!wordsFile) {

    lcd.setCursor(0,1); lcd.print(F("words.txt?"));

    while (1) { }

  }

  // Index and shuffle

  if (!indexWords()) {

    lcd.setCursor(0,1); lcd.print(F("No words found"));

    while (1) { }

  }

  // Seed RNG on boot (extra entropy later on first button)

  randomSeed(analogRead(A5)); // floating analog adds some noise

  beep\_power\_on();

  String two = format\_for\_lcd("Press Start");

  showScoresAndText(two);

  gameState = READY;

  score\_team1 = 0; score\_team2 = 0;

}

void loop() {

  btnStart.update();

  btnT1.update();

  btnT2.update();

  btnNext.update();

  btnMute.update();

  if (!rng\_seeded && (btnStart.justPressed() || btnT1.justPressed() || btnT2.justPressed() || btnNext.justPressed() || btnMute.justPressed())) {

    randomSeed(micros());

    rng\_seeded = true;

  }

  if (btnMute.justPressed()) {

    muted = !muted;

    lcdClearLine(1);

    lcd.setCursor(4,1);

    lcd.print(muted ? F("Muted") : F("Sound On"));

    if (!muted) beep\_small();

    delay(300);

    if (gameState == IN\_ROUND) showWord(currentWord);

    else {

      String two = format\_for\_lcd("Press Start");

      showScoresAndText(two);

    }

  }

  switch (gameState) {

    case READY:

      if (btnStart.justPressed()) startRound();

      if (btnT1.justPressed()) {

        score\_team1++; beep\_small();

        if (score\_team1 == 7) { lcd.clear(); lcd.setCursor(0,0); lcd.print(F("Team1 Wins!")); gameState = GAME\_DONE; }

        else { String two = format\_for\_lcd("Press Start");

        showScoresAndText(two); }

      }

      if (btnT2.justPressed()) {

        score\_team2++; beep\_small();

        if (score\_team2 == 7) { lcd.clear(); lcd.setCursor(0,0); lcd.print(F("Team2 Wins!")); gameState = GAME\_DONE; }

        else { String two = format\_for\_lcd("Press Start"); showScoresAndText(two); }

      }

      break;

    case IN\_ROUND:

      if (btnStart.justPressed()) { endRound(false); break; } // stop early

      if (btnNext.justPressed()) {

        if (getNextRandomWord(currentWord)) showWord(currentWord);

      }

      do\_tic\_toc();

      break;

    case GAME\_DONE:

      if (btnStart.justPressed()) {

        score\_team1 = 0; score\_team2 = 0;

        String two = format\_for\_lcd("Press Start");

        showScoresAndText(two);

        gameState = READY;

      }

      break;

  }

  delay(5);

}