#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

#include <Arduino.h>

#include <DFRobotDFPlayerMini.h>

#include <WiFi.h>

#include <NTPClient.h>

#include <WiFiUdp.h>

#include <FirebaseESP32.h>

#include <esp\_now.h>

#include <TimeLib.h>

// I2C LCD configuration

LiquidCrystal\_I2C lcd(0x27, 20, 4); // Address 0x27

// Ultrasonic sensor pins

#define TRIG\_PIN 4

#define ECHO\_PIN 2

// Push buttons for input

#define BUTTON\_HASH 12 // Pin for YES button

#define BUTTON\_STAR 13 // Pin for NO button

// Audio player serial configuration

HardwareSerial mySerial(1);

DFRobotDFPlayerMini myDFPlayer;

// Constants for distance detection

const unsigned long interval = 15000; // 15 seconds in milliseconds

unsigned long startMillis;

bool objectDetected = false;

bool isPlaying = false;

bool measureDistance = false;

int previousLaserState = -1; // To store previous laser state

// WiFi and NTP client configuration

const char\* ssid = "Pixel\_6pro";

const char\* password = "12345687";

WiFiUDP ntpUDP;

NTPClient timeClient(ntpUDP, "pool.ntp.org", 19800, 60000); // Offset for Sri Lanka (UTC +5:30)

// Firebase credentials

#define FIREBASE\_HOST "https://l1project-681ca-default-rtdb.firebaseio.com"

#define FIREBASE\_API\_KEY "AIzaSyDPwA\_y2ZVycvIoek0bmrobve6BT1G-FZ0"

FirebaseData firebaseData;

FirebaseConfig firebaseConfig;

FirebaseAuth firebaseAuth;

int recordCounter = 0;

void setup() {

  lcd.init();

  lcd.backlight();

  lcd.print("PRACTICE MAKES SKILL");

  lcd.setCursor(0, 1);

  lcd.print(" ");

  lcd.setCursor(0, 2);

  delay(10000);

  // Initialize serial communication

  Serial.begin(115200);

  // Configure ultrasonic sensor pins

  pinMode(TRIG\_PIN, OUTPUT);

  pinMode(ECHO\_PIN, INPUT);

  // Configure button pins

  pinMode(BUTTON\_HASH, INPUT\_PULLUP);

  pinMode(BUTTON\_STAR, INPUT\_PULLUP);

  // Initialize the DFPlayer Mini

  mySerial.begin(9600, SERIAL\_8N1, 18, 19);

  Serial.println("Initializing DFPlayer Mini...");

  if (!myDFPlayer.begin(mySerial)) {

    Serial.println("Unable to begin:");

    Serial.println("1. Please recheck the connection!");

    Serial.println("2. Please insert the SD card!");

    while (true);

  }

  Serial.println(F("DFPlayer Mini online."));

  myDFPlayer.setTimeOut(500); // Set serial communication timeout to 500ms

  myDFPlayer.volume(30); // Set volume (0 to 30)

  startMillis = millis(); // Initialize startMillis

  // Connect to Wi-Fi

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL\_CONNECTED) {

    delay(1000);

    Serial.println("Connecting to WiFi...");

  }

  Serial.println("Connected to WiFi");

  // Initialize NTP client

  timeClient.begin();

  timeClient.update();

  // Initialize Firebase

  firebaseConfig.host = FIREBASE\_HOST;

  firebaseConfig.api\_key = FIREBASE\_API\_KEY;

  firebaseAuth.user.email = "224094@example.com";

  firebaseAuth.user.password = "amma1234";

  Firebase.begin(&firebaseConfig, &firebaseAuth);

  Firebase.reconnectWiFi(true);

  // Initialize the record counter if it doesn't exist

  if (!Firebase.getInt(firebaseData, "/runner/recordCounter")) {

    Firebase.setInt(firebaseData, "/runner/recordCounter", 0);

  } else {

    recordCounter = firebaseData.intData();

  }

}

void loop() {

  // Check laser state from Firebase

  if (!Firebase.getInt(firebaseData, "/laserstate")) {

    Serial.print("Error getting laserstate: ");

    Serial.println(firebaseData.errorReason());

  } else {

    int laserState = firebaseData.intData();

    if (laserState != previousLaserState) {

      previousLaserState = laserState;

      if (laserState == 1) {

        clearRow(2);

        lcd.print("Are You Ready?");

        lcd.setCursor(0, 3);

      } else {

        clearRow(2);

        lcd.print("Laser Not Detected");

        lcd.setCursor(0, 3);

      }

    }

  }

  if (previousLaserState == 1 && !measureDistance) {

    if (digitalRead(BUTTON\_HASH) == LOW) {

      measureDistance = true; // Start measuring distance

      lcd.clear();

      lcd.print("Place on Position,");

      lcd.setCursor(0, 1);

      lcd.print("Command will Play");

      lcd.setCursor(0, 2);

      lcd.print("Soon");

      lcd.setCursor(0, 3);

      delay(10000);0

      lcd.clear();

    } else if (digitalRead(BUTTON\_STAR) == LOW) {

      measureDistance = false; // Stop measuring distance

      Serial.print("Answer is NO");

      lcd.clear();

      lcd.print("TAKE TIME & BE READY");

      lcd.setCursor(0, 1);

      delay(10000);

      lcd.print("");

      lcd.setCursor(0, 2);

      lcd.print("Are You Ready?");

      lcd.setCursor(0, 3);

    }

  }

  if (measureDistance && !isPlaying) {

    long duration, distance;

    unsigned long currentMillis = millis();

    // Send a pulse to trigger the ultrasonic sensor

    digitalWrite(TRIG\_PIN, LOW);

    delayMicroseconds(2);

    digitalWrite(TRIG\_PIN, HIGH);

    delayMicroseconds(10);

    digitalWrite(TRIG\_PIN, LOW);

    // Read the duration of the echo

    duration = pulseIn(ECHO\_PIN, HIGH);

    distance = duration \* 0.034 / 2; // Calculate the distance in cm

    // Print distance to Serial Monitor

    Serial.print("Distance: ");

    Serial.print(distance);

    Serial.println(" cm");

    // Check if the distance is within 200 cm

    if (distance <= 200) {

      if (!objectDetected) {

        objectDetected = true;

        startMillis = currentMillis; // Record the start time if an object is detected

      }

      // Check if the time interval has passed to trigger audio playback

      if (currentMillis - startMillis >= interval) {

        // Start playing audio if not already playing

        if (!isPlaying) {

          myDFPlayer.play(1); // Play the first track

          isPlaying = true;

          // Get current date and time

          time\_t rawTime = timeClient.getEpochTime();

          struct tm\* timeInfo = localtime(&rawTime);

          // Format the current date and time into a string

          char currentDate[11];

          char currentTime[30];

          sprintf(currentDate, "%04d-%02d-%02d",

                  timeInfo->tm\_year + 1900, timeInfo->tm\_mon + 1, timeInfo->tm\_mday);

          sprintf(currentTime, "%04d-%02d-%02d %02d:%02d:%02d:%03d",

                  timeInfo->tm\_year + 1900, timeInfo->tm\_mon + 1, timeInfo->tm\_mday,

                  timeInfo->tm\_hour, timeInfo->tm\_min, timeInfo->tm\_sec,

                  millis() % 1000); // Get the current milliseconds

          // Calculate the new time by adding 9.030 seconds

          time\_t newRawTime = rawTime + 9; // Add 9 seconds

          int newMillis = (millis() % 1000) + 30; // Add 30 milliseconds

          // Handle millisecond overflow

          if (newMillis >= 1000) {

            newRawTime += 1;

            newMillis -= 1000;

          }

          struct tm\* newTimeInfo = localtime(&newRawTime);

          // Format the new time into a string

          char newTime[30];

          sprintf(newTime, "%02d:%02d:%02d:%03d",

                  newTimeInfo->tm\_hour, newTimeInfo->tm\_min, newTimeInfo->tm\_sec,

                  newMillis);

          // Print the current date and time

          Serial.print("Current date: ");

          Serial.println(currentDate);

          Serial.print("Current time: ");

          Serial.println(currentTime);

          // Print the new time

          Serial.print("New time: ");

          Serial.println(newTime);

          // Increment the record counter

          recordCounter++;

          // Update the record counter in Firebase

          if (!Firebase.setInt(firebaseData, "/runner/recordCounter", recordCounter)) {

            Serial.print("Error setting recordCounter: ");

            Serial.println(firebaseData.errorReason());

          }

          // Store the current date and new time in Firebase under the new record

          String recordPath = "/runner/" + String(recordCounter);

          if (Firebase.setString(firebaseData, recordPath + "/Date", currentDate) &&

              Firebase.setString(firebaseData, recordPath + "/Starttime", newTime)) {

            Serial.println("Date and time recorded successfully.");

          } else {

            Serial.print("Error recording date and time: ");

            Serial.println(firebaseData.errorReason());

          }

          delay(30000);

          myDFPlayer.stop();

          // Reset the board after stopping the player

          ESP.restart();

        }

      }

    } else {

      // No object detected, reset state

      objectDetected = false;

      isPlaying = false;

    }

  }

}

void clearRow(int row) {

  lcd.setCursor(0, row);  // Move cursor to the beginning of the specified row

  for (int i = 0; i < 20; i++) {  // Assuming a 20-character wide display

    lcd.print(" ");  // Print spaces to clear the row

  }

  lcd.setCursor(0, row);  // Move cursor back to the beginning of the cleared row

}