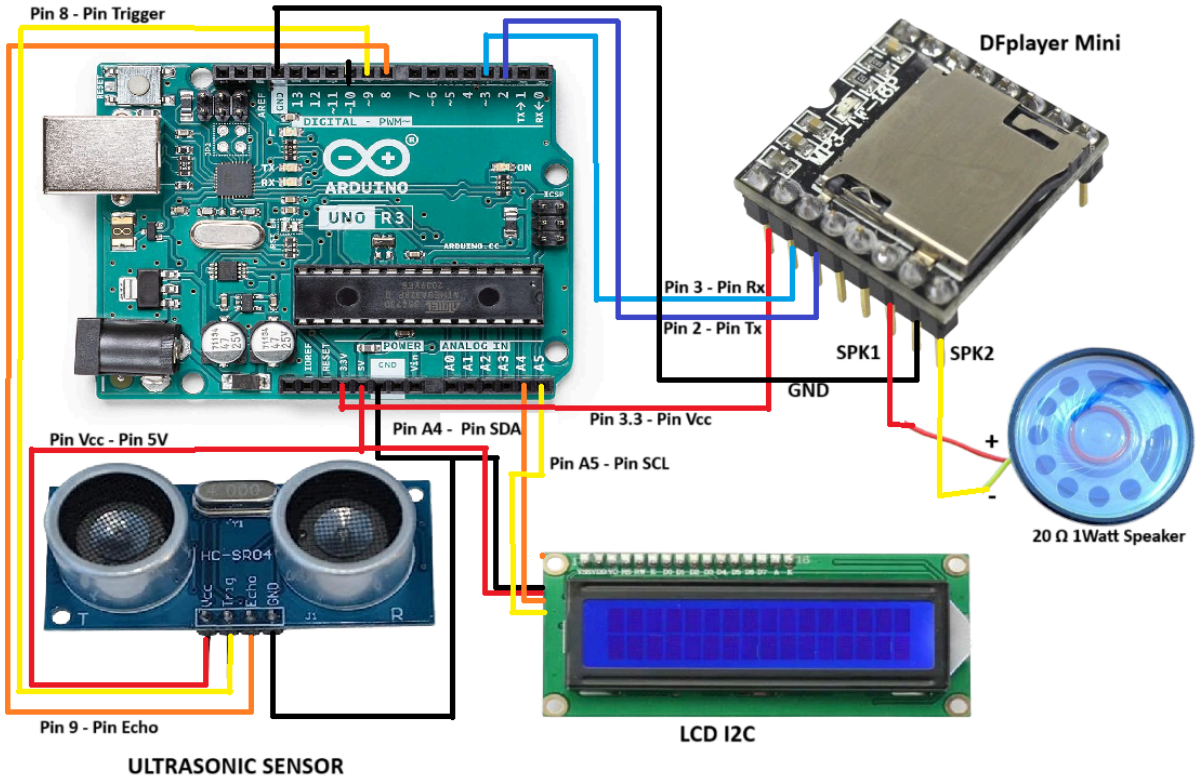
Components used

1- Arduino Uno R3
2- DFPlayer Mini
3- Ultrasonic Sensor
4- 20Ohm 1Watt Speaker
5- LCD Display(I2C)

Circuit Diagram



Arduino Code

```
#include <LiquidCrystal_I2C.h>
#include <HCSR04.h>
#include "DFRobotDFPlayerMini.h"
#include "SoftwareSerial.h"

// Pin definitions for the ultrasonic sensor
```

```
#define TRIGGER_PIN 9
#define ECHO_PIN 8

// Global variables
double distance_cm = 0, depth = 0;
double reference_distance = -1; // Reference distance when the cup is
empty

// Constants
const long TRACK_INTERVAL = 2000;         // Interval for track playback
(ms)
const long SENSOR_INTERVAL = 500;        // Interval for ultrasonic sensor
measurements (ms)
const long LCD_UPDATE_INTERVAL = 400;    // Interval for LCD updates (ms)
const double MAX_DISTANCE = 400;         // Maximum valid distance in cm

// Hardware initialization
UltraSonicDistanceSensor distanceSensor(TRIGGER_PIN, ECHO_PIN);
LiquidCrystal_I2C lcd(0x27, 16, 2);
SoftwareSerial mySoftwareSerial(2, 3); // RX, TX for DFPlayer Mini
DFRobotDFPlayerMini myDFPlayer;

// Timing variables
unsigned long previousTrackMillis = 0;
unsigned long previousSensorMillis = 0;
unsigned long previousLCDMillis = 0;
bool trackPlayed = false;

void setup() {
  Serial.begin(115200);                    // Initialize Serial Monitor
  mySoftwareSerial.begin(9600);            // Initialize SoftwareSerial
for DFPlayer Mini

  lcd.init();                              // Initialize LCD
  lcd.backlight();                         // Enable LCD backlight

  // Wait for Serial communication to initialize
  while (!Serial);

  // Initialize DFPlayer Mini
```

```
  if (!myDFPlayer.begin(mySoftwareSerial)) {
    Serial.println(F("DFPlayer initialization failed!"));
    Serial.println(F("1. Check connections."));
    Serial.println(F("2. Insert an SD card."));
    while (true);
  }

  Serial.println(F("DFPlayer Mini initialized!"));
  myDFPlayer.setTimeOut(500);               // Set timeout for DFPlayer
communication
  myDFPlayer.volume(30);                    // Set DFPlayer volume (0-30)
  myDFPlayer.EQ(0);                         // Set EQ mode (0: Normal)
}

void loop() {
  unsigned long currentMillis = millis();  // Get current time

  // Measure distance at SENSOR_INTERVAL
  if (currentMillis - previousSensorMillis >= SENSOR_INTERVAL) {
    previousSensorMillis = currentMillis;
    distance_cm = distanceSensor.measureDistanceCm();

    // Validate distance
    if (distance_cm <= 0 || distance_cm > MAX_DISTANCE) {
      distance_cm = 0; // Ignore invalid readings
    }

    // Log valid measured distance
    Serial.print("Measured Distance: ");
    Serial.println(distance_cm);

    // Set reference distance if not already set
    if (reference_distance < 0 && distance_cm > 0) {
      reference_distance = distance_cm;
      Serial.print("Reference Distance Set: ");
      Serial.println(reference_distance);
    }
  }

  // Calculate depth if reference and distance are valid
```

```
  if (reference_distance > 0 && distance_cm > 0) {
    depth = reference_distance - distance_cm;
    if (depth < 0) depth = 0; // Prevent negative depth
  } else {
    depth = 0; // Reset depth if invalid
  }

  // Play a track at TRACK_INTERVAL
  if (currentMillis - previousTrackMillis >= TRACK_INTERVAL) {
    previousTrackMillis = currentMillis;
    playTrack(depth);
    trackPlayed = true;
  }

  // Update the LCD at LCD_UPDATE_INTERVAL
  if (currentMillis - previousLCDMillis >= LCD_UPDATE_INTERVAL) {
    previousLCDMillis = currentMillis;
    updateLCD(depth);
  }
}

// Function to play tracks based on depth
void playTrack(double depth) {
  if (depth >= 1.00 && depth <= 1.99) {
    myDFPlayer.play(1); // Play track 1
  } else if (depth >= 2.00 && depth <= 2.99) {
    myDFPlayer.play(2); // Play track 1
  } else if (depth >= 3.00 && depth <= 3.99) {
    myDFPlayer.play(3); // Play track 2
  } else if (depth >= 4.00 && depth <= 4.99) {
    myDFPlayer.play(4); // Play track 3
  } else if (depth >= 5.00 && depth <= 5.99) {
    myDFPlayer.play(5); // Play track 4
  }
}

// Function to update the LCD with depth
void updateLCD(double depth) {
  lcd.clear();
  lcd.setCursor(0, 0);
```

```
  lcd.print("Depth:");
  lcd.setCursor(0, 1);
  if (depth > 0) {
    lcd.print(depth, 2); // Display depth with 2 decimal places
  } else {
    lcd.print("Invalid");
  }
}
```

You cand Find the code, circuit diagram and the audios here:

https://github.com/Valentim-bot/Water-Level-Measurement-with-Audio-Arduino-Ultrasonic-Sensor-DFPlayer-Mini.git

How to put the audios inside the sd card?

1- Your micro SD card should support FAT16 and FAT32 file systems

2 - Ordinary folders must be renamed as 01, 02, 03.....99, and the audio files must be renamed as
001.mp3/001.wav, 002.mp3/002.wav, 003.mp3/003.wav, ...255.mp3/255.wav. It is also possible to keep the
original name when you rename a file. For example, the original name is “Yesterday Once More.mp3”, then you can
rename it as “001Yesterday Once More.mp3”.