Magic 8 Ball

Autor: Poașcă Ștefan
Grupa: 333CC

Introducere

Prin acest proiect doresc sa replic modul de functionare a unei jucarii foarte populare, Magic 8 Ball, aceasta constand intr-o bila neagra, similara ca si aspect cu una de biliard.
Modul de functionare este destul de simplu : ii pui o intrebare bilei, apoi o agiti, iar aceasta va genera un raspuns random dintr-un set de 15 raspunsuri : 5 afirmative, 5 negative si 5 nesigure.
Fiind o replica a unei jucarii, are scopul de a distra lumea si de a raspunde intr-un mod aleator la intrebarile utilizatorului.

Descriere generală

Utilizatorul va pune o intrebare (input pentru microfon) si va agita proiectul (input pentru accelerometru).
Microfonul si accelerometrul dau input-ul primit de la utilizator ca input pentru Arduino Uno.
Arduino Uno foloseste ca output LCD-ul pentru feedback vizual si buzzer-ul pentru feedback auditiv.
De asemenea, LCD-ul primeste ca input pentru pinul V0 output-ul potentiometrului.

Hardware Design

Folosesc urmatoarele componente :

- Placa Arduino Uno
- FC-04 Sound Sensor Module

- 1602 LCD with Blue Backlight

- MPU6050 Accelerometer and Gyroscope Module

- 10K Potentiometer

- 5V Passive Buzzer

Software Design

- **Mediu de dezvoltare:** Arduino IDE
- **Librarii si surse 3rd-party:** Wire.h, LiquidCrystal.h, Adafruit_MPU6050.h, pitches.h

O sa prezint implementarea prin impartirea programului dupa componentele hardware folosite :

- **Display**
  - Folosesc libraria LiquidCrystal.h pentru a initializa obiectul “lcd” care reprezinta modul prin care o sa interactionez cu LDC-ul fizic
  - Am initializat “lcd” cu contructorul pentru 4 pini de date (5, 4, 3, 2), 1 pin de Register Select (12) si unul de Enable (11)
  - Am facut un array care contine 15 raspunsuri posibile de afisat : 5 pozitive, 5 negative si 5 nesigure
  - Pentru a nu avea cuvinte care se sfarsez “abrupt” pe primul rand si se continua pe al doilea rand, am scris functia **print_on_2_line** care se asigura ca, daca am un cuvant care este fragmentat la sfarsitul liniei, o sa il scrie pe linia urmatoare
- **Accelerometru**
  - Am intializat obiectul de tip Adafruit_MPU6050 pentru a ma putea folosi de API-ul oferit
  - Am setat limitele ca si intensitate si durata pentru detectia miscarii dupa mai multe incercari unde am determinat valorile optime pentru proiectul meu
  - Am activat detectia miscarilor dupa parametrii setati
- **Buzzer**
  - Am cautat si gasit libraria care face conexiunea intre frecventele buzzer-ului si notele muzicale canonice
  - Am scris 3 functii care fac buzzer-ul sa cante in 3 moduri diferite 3 cantece corespunzatoare tipurilor de raspunsuri

Am facut algoritmul asemanator cu al unui joc, in care programul se afla mereu intr-o anumita stare din cele 5 definite de mine. De mentionat faptul ca prima rulare cu un status nou afiseaza mesajul corespunzator pe LCD, urmand ca de la a 2-a rulare sa interactionez cu senzorii (vezi sectiunea de optimizari) :

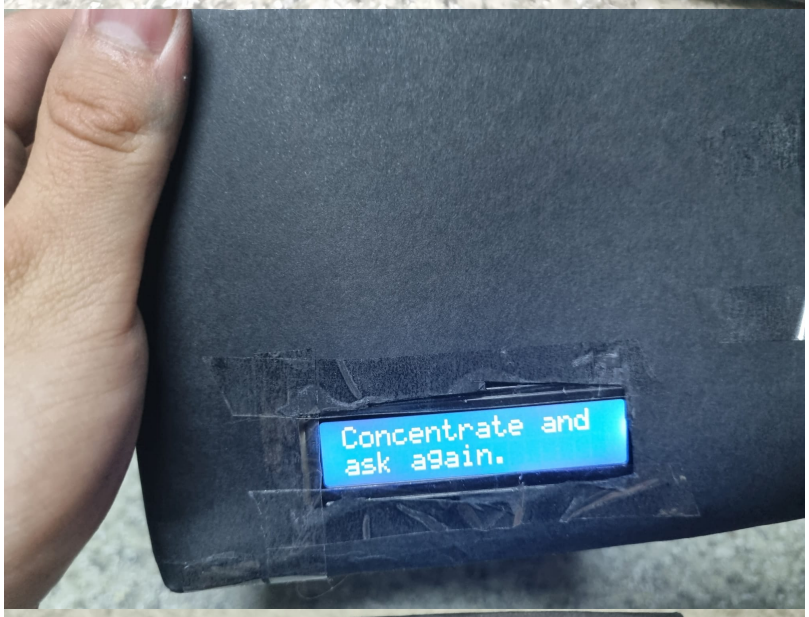
- **STATUS_WAITING_QUESTION** : Starea in care initiez jocul. Astept un input de la microfon care depaseste 100 (nu este zgomot de fundal) pe pinul conectat (A0). Daca este detectat, se trece in urmatoarea stare.
- **STATUS_DURING_QUESTION** : In aceasta stare astept pana cand persoana se opreste din vorbit, prin resetare timer-ului asociat acestei stari de fiecare data cand detectez un sunet care trece de 100 (persoana inca vorbeste). Daca timer-ul ajunge la valoarea stabilita, persoana nu a mai vorbit un interval de secunde suficient cat sa presupun ca a terminat de pus intrebarea si se trece la urmatoarea stare.
- **STATUS_WAITING_SHAKE** : Dupa ce persoana a pus intrebarea, astept ca accelerometrul sa

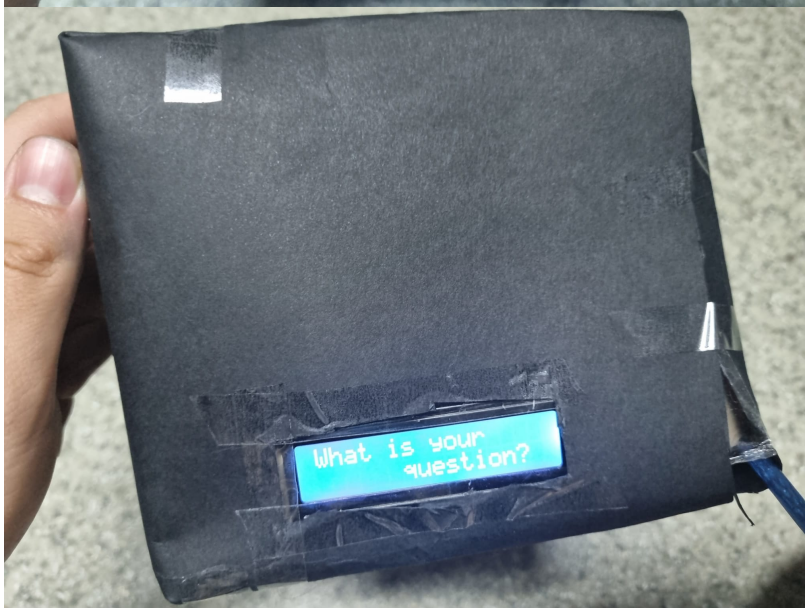
genereze un MotionInterrupt la detectarea unei miscari care sa satisfaca pragurile setate initial.
- **STATUS_DURING_SHAKE** : Similar cu **STATUS_DURING_QUESTION**, astept terminarea detectiilor de miscare, iar daca detectez una, resetez timer-ul asociat.
- **STATUS_SHOW_RESPONSE** : Starea finala. Dupa ce nu s-a mai detectat miscare, se asuma ca persoana a incetat agitarea aparatului, deci se poate afisa raspunsul la intrebare, impreuna cu melodia corespunzatoare tipului de mesaj rezultat. Dupa terminarea tonului, se revine la **STATUS_WAITING_QUESTION**

Alte mentiuni / optimizari:

- Nu am fost multumit de rezultatele obtinute din punctul de vedere al randomness-ului atunci cand generam o valoare aleatorie doar cand aveam nevoie de aceasta (dispersia valorilor era destul de slaba). Am rezolvat problema folosind ca seed o valoare citita “din vid” : un **analogRead la un pin neconectat** (aceasta schimbandu-se la fiecare rulare), precum si prin faptul ca in fiecare loop() generez o valoare aleatorie, si atunci cand am nevoie cu adevarat de una, salvez “instanta” valorii din acel ciclu
- Am optimizat folosirea memoriei prin 2 metode:
  - Pentru string-urile pe care le folosesc doar o data, am utilizat macro-ul F() prin care i se comunica compilatorului ca string-ul se va salva in PROGMEM, nu in RAM
  - Pentru string-urile refolosibile, am salvat intreaga “banca” de raspunsuri in **PROGMEM** utilizand cuvantul-cheie asociat, apoi am folosit functiile strcpy_P si pgm_read_word pentru a transfera doar raspunsul dorit din program space in RAM.
- Pentru a optimiza numarul de scrieri pe care le fac pe LCD (si pentru a **evita efectul de “flickering”** rezultat), am gandit algoritmul astfel incat afisarea sa se faca doar o data per etapa, urmand ca la urmatoarea rulare a loop(), executabilul sa intre pe o alta ramura in care nu se va mai face nicio afisare

Rezultate Obţinute


Concentrate and
ask again.


What is your
question?

Concluzii

Download

Download source code

Bibliografie/Resurse

Datasheet MPU-6050
Datasheet 1602 LCD

Interfacing FC-04 Microphone Sound Sensor Module with Arduino
Interfacing 16×2 Character LCD Module with Arduino
How to Program MPU 6050 With Arduino
Piezo Speaker with Arduino

Export to PDF