JOC "MAGIC CAST" - BAGHETĂ MAGICĂ

Introducere

Proiectul presupune realizarea unei “baghete magice” prin intermediul căreia se va putea juca un minigame. Bagheta va:

- primi date de poziționare în spațiu de la un accelerometru
- calcula prin anumite formule geometrice matematice și metode de atenuare a zgomotului forma geometrică realizată (ex: cerc, pătrat, zig-zag)
- va dispune de diferite ”vrăji„ depinzând de forma realizată (exemplu: cerc → apă; pătrat → pământ)
- va avea efecte tipice vrăjii realizate (led, buzzer)
- (posibil) va avea output pe monitorul unui laptop unde va rula un cod de minigame în care diferiți monștrii vor fi uciși cu diferite vrăji

Ideea de bază este realizarea unui joc simpu și intuitiv pe care colegii să-l poată juca la târgul de proiecte PM.

Descriere generală

Descrierea Modulelor și a Modului de Interacțiune

Proiectul “Magic Cast” este compus dintr-o serie de module hardware (senzori, actuatori, unitate de procesare) și module software (algoritmi de procesare a semnalului și decizie), care colaborează pentru a transforma mișcarea fizică într-o acțiune virtuală.

1. Module Hardware

- Sursă de Alimentare (Acumulator 18650 + Modul Boost 5V): Asigură portabilitatea “baghetei”. Modulul de boost ridică tensiunea bateriei Li-Ion la 5V stabili, alimentând direct microcontrolerul și senzorul.
- Modul Input Senzorial (MPU6050 - Accelerometru & Giroscop): Este responsabil cu captarea datelor de mișcare pe 6 axe (accelerație și rotație) în spațiul 3D.
- Buton “Trăgaci” (Push-button): Un element de control conectat la un pin digital (GPIO) cu rezistență de pull-up internă. Acționează ca un semnal de declanșare (trigger) pentru a delimita momentul în care utilizatorul desenează forma în aer.
- Unitatea Centrală (Microcontroler ATmega328P - Xplained Mini): Reprezintă creierul hardware al proiectului. Acesta interoghează senzorii, execută algoritmul software și comandă perifericele de ieșire.
- Module de Feedback Local (Buzzer Pasiv & LED RGB): Oferă confirmare senzorială utilizatorului.

Buzzerul generează feedback audio (frecvențe specifice fiecărei vrăji prin semnale PWM), iar LED-ul RGB își schimbă culoarea în funcție de elementul vrăjii invocate (ex: albastru pentru apă, roșu pentru foc).
- Interfața cu Jocul (Laptop / Monitor): Rulează aplicația/jocul final care primește comenzile și afișează efectele vizuale (ex: distrugerea monștrilor).

2. Module Software (Logica internă pe ATmega328P)

- Modulul de Filtrare Zgomot: Preia datele brute de la senzorul MPU6050, care conțin adesea "zgomot" cauzat de tremurul mâinii. Folosește un filtru matematic (ex: filtru trece-jos sau filtru complementar) pentru a netezi valorile înainte de analiză.
- Modulul de Detecție Traseu (Segmentare): Acest modul software se activează doar când starea butonului hardware trece în LOW (apăsat). Înregistrează coordonatele mișcării succesive într-un buffer (vector de date) pentru a forma un traseu continuu.
- Modulul de Identificare Vrajă (Pattern Matching): Procesează traseul finalizat (după eliberarea butonului). Analizează variațiile geometrice din buffer și, printr-un algoritm de recunoaștere a formelor (ex: calcularea schimbărilor de direcție sau distanțe minime), asociază traseul cu un anumit gest prestabilit (Cerc, Pătrat, Zig-Zag).

3. Modul de interacțiune a modulelor (Fluxul datelor) Sistemul funcționează pe baza unui flux continuu de tipul Input → Procesare → Output:

- Inițierea acțiunii: Microcontrolerul așteaptă o acțiune pe pinul GPIO al Butonului Trăgaci. Când utilizatorul apasă butonul și mișcă bagheta, sistemul intră în starea de achiziție.
- Achiziția datelor: Prin intermediul magistralei I2C (pinii SDA și SCL), ATmega328P citește în mod repetat datele senzorului MPU6050 la o frecvență fixă.
- Procesarea (Software): Datele citite trec secvențial prin Modulul de Filtrare, sunt stocate de Modulul de Detecție a Traseului, iar la eliberarea butonului, Modulul de Identificare analizează traseul și returnează un cod specific vrăjii (ex: VRAJA_CERC).
- Generarea Output-ului: Odată ce vraja a fost identificată, microcontrolerul ia trei acțiuni simultane:
  - Trimite semnale PWM modificate către Buzzer pentru a genera sunetul specific.
  - Modifică starea pinilor PWM/GPIO legați la LED-ul RGB pentru a afișa culoarea vrăjii.
  - Transmite un șir de caractere (ex: "CAST:WATER\n") prin protocolul UART (Serial) către Laptop, declanșând acțiunea în jocul video.

Hardware Design

Lista de piese (Bill of Materials - BOM)

- Microcontroler: Placă de dezvoltare ATmega328P Xplained Mini (creierul proiectului).
- Senzor Mișcare: Modul GY-521 (IC original MPU-6050) – Accelerometru și giroscop pe 3 axe (comunicație I2C).
- Alimentare: Carcasă Powerbank USB 5V pentru acumulatori 18650 – Va oferi tensiune stabilă de 5V și portabilitate pentru baghetă.
- Buton Control: Buton Push 6x6x6 mm – Folosit ca "trăgaci" pentru a iniția citirea gestului.
- Feedback Audio: Buzzer pasiv (difuzor intern de placă de bază PC) – Pentru redarea efectelor sonore specifice fiecărei vrăji.
- Feedback Vizual: * 1x LED RGB 10mm (catod comun) – Va lumina în culoarea specifică vrăjii (ex:

roșu=foc, albastru=apă).
- * LED-uri opționale pentru status (Bicolor, Galben 3mm, Verde 5mm) – Pentru a semnaliza starea baghetei (ex: galben pentru ”mod înregistrare”, verde pentru ”gata de utilizare”).
- Unordered List ItemSuport Fizic: Placă PCB prototipare dublă față, 4x6cm, FR4.

Pin description:

- Modul GY-521 (Accelerometru/Giroscop): Conectat la pinii PC4 (SDA) si PC5 (SCL). Foloseste interfata I2C nativa pentru citirea datelor de miscare.
- LED RGB (Catod Comun): Conectat la pinii PD3, PD4 si PD5 (prin rezistente de 220 ohmi). Pinii genereaza semnale PWM pentru a controla individual culorile Rosu, Verde si Albastru.
- Buton Control: Conectat la pinul PD6. Pinul este setat ca intrare digitala cu rezistenta interna activata (INPUT_PULLUP) si face contact la GND in momentul apasarii.
- Buzzer Pasiv: Conectat la pinul PD7. Este utilizat pentru a trimite semnale oscilante din cod pentru generarea notelor muzicale.
- Alimentare (VCC si GND): Pinii de 5V si GND ai placii. Toate componentele impart un nod de masa comun pe placa de prototipare.

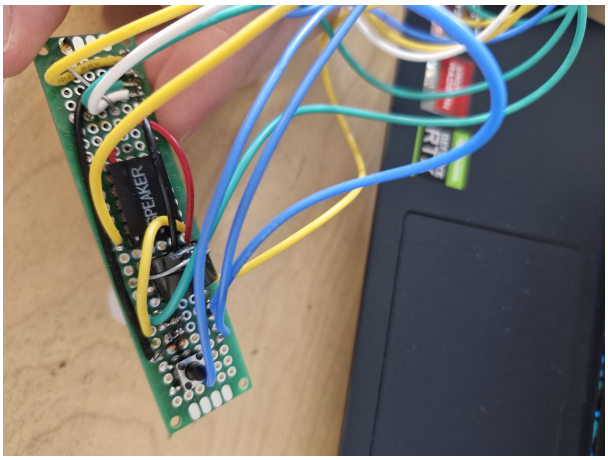
1. Diagrama de semnal I2C (Comunicația cu senzorul MPU-6050)

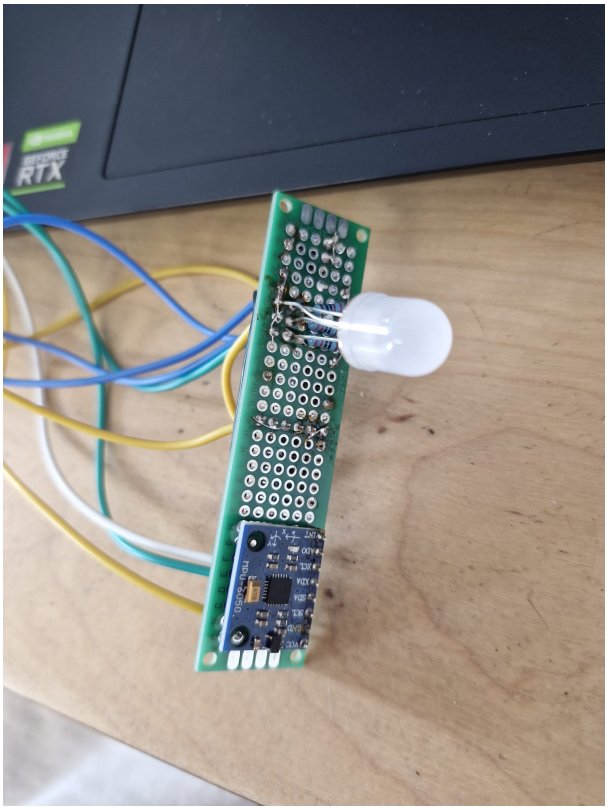
Comunicația este inițiată printr-o condiție de START pe linia de date (SDA), urmată de transmiterea adresei senzorului și a bitului de Read/Write. Sincronizarea este menținută de semnalul de ceas (SCL) generat de microcontroler. Fiecare pachet de date este validat de un bit de confirmare (ACK).

2. Diagrama de semnal CTC / Timer (Controlul Buzzerului Pasiv)

Pentru a crea efectele sonore specifice fiecărei vrăji (frecvențe diferite), se utilizează Timer/Counter-ul intern al microcontrolerului în modul CTC (Clear Timer on Compare Match). Conform diagramei, timerul (TCNTn) se incrementează sincron cu ceasul (divizat prin prescaler) până când atinge valoarea maximă definită în registrul de comparație (OCRnx). În acel moment (TOP), timerul se resetează la zero și flag-ul de Output Compare (OCFnx) este setat. Poate fi controlat software exact prin modificarea valorii TOP, dictând astfel nota muzicală redată de buzzer.

3. Piesele Asamblate si Videoclipuri





1. Functionalitate LED, BUZZER, BUTON:

```
 Ledul face switch intre cele 3 culori. Cand se apasa butonul, se opreste ledul si se aprinde buzzer-ul
```

https://youtube.com/shorts/iP0EWmqF4O8

2. Functionalitate SENZOR MISCARE

```
 Giroscopul transmite coordonatele x,y,z ( grade/secunda rotatie in jurul axelor proprii - x, y sau z) si niste coordonate extra (temperatura).
 Accelerometrul trimite coordonatele x,y,z (forta/secunda forta de miscare aplicata pe axele proprii - x, y sau z)
```

Miscarea este detectata https://youtu.be/p2cHBI9OktE

Software Design

1. Mediu de dezvoltare:

Arduino IDE, C++

2. Librării și surse 3rd-party

- Wire.h (librărie standard) - configurarea și realizarea comunicației I2C între microcontroller și senzorul MPU6050.

- Arduino.h (librărie standard) - accesul la funcțiile hardware de bază (GPIO, PWM, timere, generare frecvențe audio).
- Librării 3rd-party externe - Teleplot pentru plotarea in timp real a graficelor datelor de miscare (necesare la descompunerea si generalizarea miscarilor si gesturilor alese).
- (citirea și configurarea senzorului au fost realizate direct la nivel de registru)

4. Limitari Hardware Accelerometrul si Giroscopul nu ofera pozitia in spatiu. piesa nu poate deduce de la sine locatia intr-o camera, inaltimea, cate grade a fost rotita sau daca a facut un cerc complet. Orice tip de miscare este transmisa drept raw data. (Ex: Forta X: 6301, Forta Y: 14114, Forta Z: 9015, Rotatie X: 7337, Rotatie Y: 100, Rotatie Z: 4315) Astfel, deducerea pozitiei in spatiu devine aproape imposibila, intrucat necesita o dubla integrare pentru fiecare tick de date primit. Asadar, ar duce la delayuri intre primirea datelor si efectuarea calculelor, iar erori de 0.02 de calcul ar fi fatale.

Asadar, am decis sa abordez o metoda de rezolvare “pattern matching” unde vizualizez trendul fiecarei miscari si ii definesc niste criterii de detectare. Obiectivele sunt SIMPLITATE in calcul si MARGGINE DE EROARE MINIMALA.

5. Abordare Experimentala Am observat limitarile vizualizarii datelor brute si am realizat un fisier test.cpp pentru a inregistra in format grafic tendintele si aspectul vizual al fiecarei miscari implementate. Am folosit un sample pool de 10-15 miscari pentru fiecare gest in parte si am realizat prin analiza lor un pattern de recunoastere. Obiectivul principal este detectara eficienta si evitarea confuziei cu alte forme / detectarea unor forme haotice ca forme intentionate.

(exemplu de date pentru o miscare de tip FLICK)

Am dedus initial ca GY-521 MPU-6050 era setat pe detectia minimala de forta: 2g (= forta gravitationala) si am marit-o pentru a nu avea in date plafonare (pierdere importanta de date).

Am marit viteza de transmisie a datelor pentru a avea un grafic cu flow eficient si am folosit graficele pt a ajusta filtrul de date.

Click pentru test.cpp

```
#include <Arduino.h>
#include <Wire.h>
#define PIN_BUTON 6
const int MPU_ADDR = 0x68;
int16_t AccelX, AccelY, AccelZ, GyroX, GyroY, GyroZ;

void setup() {
  Serial.begin(38400);
  Wire.begin();
  pinMode(PIN_BUTON, INPUT_PULLUP);
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);
  //filtru DLPF(Treapta 4=~20Hz)
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(0x1A);
```

```
  Wire.write(0x04);
  Wire.endTransmission(true);
  // accelometru la ±8g
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(0x1C);
  Wire.write(0x10);
  Wire.endTransmission(true);
  // giroscop: ±1000 grade/sec
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(0x1B);
  Wire.write(0x10);
  Wire.endTransmission(true);
}

void loop() {
  if (digitalRead(PIN_BUTON) == LOW) {
    Wire.beginTransmission(MPU_ADDR);
    Wire.write(0x3B);
    Wire.endTransmission(false);
    Wire.requestFrom(MPU_ADDR, 14, true);

    AccelX = Wire.read() << 8 | Wire.read();
    AccelY = Wire.read() << 8 | Wire.read();
    AccelZ = Wire.read() << 8 | Wire.read();
    Wire.read(); Wire.read(); // don't need temperature
    GyroX  = Wire.read() << 8 | Wire.read();
    GyroY  = Wire.read() << 8 | Wire.read();
    GyroZ  = Wire.read() << 8 | Wire.read();
    //Teleplot format
    Serial.print(">Gyro_Y_Twist:"); Serial.println(GyroY);
    Serial.print(">Accel_X_Zgomot:"); Serial.println(AccelX);
    Serial.print(">Accel_Z_Zgomot:"); Serial.println(AccelZ);

    delay(5);
  }
}
```

6.Jurnalul Descoperirilor si Implementarilor Tehnice

1. Descompunerea Vectoriala 3D si Zgomotul Lateral (Fizica / Matematica)
   Am descoperit că o mișcare aparent unidirecțională (scuturarea baghetei) nu are loc perfect pe o singură axă. Am observat cum forța se distribuie organic pe axele adiacente (Y și Z), obținând astfel un tablou 3D al mișcării umane naturale și demonstrând imposibilitatea unei lovituri perfect drepte din încheietură.
2. Magnitudinea Vectoriala prin Teorema lui Pitagora (Matematica)
   Pentru a stabili cu precizie momentul declansarii senzorului (wake-up trigger), am aplicat geometria de baza in spatiul 3D: $\sqrt{X^2 + Y^2 + Z^2}$. Acest calcul a permis obtinerea unei valori absolute a fortei aplicate, complet independenta de orientarea spatiala sau directia baghetei.
3. Efectul de Bici si Forta Centrifuga (Fizica / Biomecanica)

Am observat pe graficele de analiza o crestere geometrica pe axa Z exact in momentul in care acceleratia pe axa X scadea brusc. Am dedus ca limitarea anatomica a incheieturii franeaza brusc miscarea liniara, transformand bagheta dintr-un corp in miscare rectilinie intr-un pendul. Aceasta observatie a validat direct principiul fizic al fortei centrifuge ($F_c = m \cdot \omega^2 \cdot r$).

4. Asimetria Biomecanica vs. Functia Modul (Fizica / Informatica)
   Am investigat motivul pentru care un gest de Flick spre dreapta (Backhand) nu genera aceleasi valori de forta ca un Flick spre stanga (Forehand). Am luat concluzia ca anatomia umana genereaza intrinsec mai putina forta la extensia spre exterior, demonstrand ca aplicarea pura a functiei matematice absolute (abs()) nu este suficienta pentru a compensa aceasta inegalitate biomecanica.
5. Analiza prin Metoda Delta / Amplitudinea Absoluta (Matematica / Procesare de Semnal)
   Pentru a rezolva limitarile directionale, am reproiectat arhitectura codului, trecand de la o evaluare instantanee la o evaluare post-actiune (declansata la eliberarea butonului). Calculand diferenta matematica maxim - minim (Delta) pe durata intregii apasari, am reusit sa captam energia cinetica totala a miscarii (energia initiala a loviturii insumata cu forta de franare/recul).
6. Semnatura Tremurului Muscular (Biomecanica / Procesare de Semnal)
   In timpul testarii gestului de Thrust (impingere), am identificat pe axa X o fluctuatie de inalta frecventa de tip “dinti de fierastrau”. Am catalogat acest tipar ca fiind vibratia mecanica inerenta a muschiului bratului tensionat la maximum. Am integrat acest “zgomot” ca filtru de excludere logica: daca tremurul pe X depaseste un anumit prag, miscarea hibrida este respinsa.
7. Saturarea Senzorului la 16-biti (Electronica / Informatica)
   La testarea gestului de Twist (rasucire), am observat plafonarea hardware a giroscopului la valoarea constanta de -32768 (limita inferioara absoluta pentru o variabila signed pe 16 biti). Am concluzionat ca viteza unghiulara generata de rotatia incheieturii atinge praguri extreme, dominand complet si facand irelevant orice zgomot liniar de fundal.
8. Arbori de Decizie Ierarhici (Informatica)
   Am implementat o logica de clasificare in cascada (if/else if) pentru diferentierea gesturilor. Am stabilit ca ordinea de evaluare a conditiilor este critica pentru eliminarea rezultatelor fals-pozitive: prioritate maxima a primit-o miscarea cu cea mai clara semnatura (Twist-ul urias pe giroscop), urmata de miscarea dominata strict de o singura axa liniara (Flick-ul lateral), lasand la final miscarea compusa, cu cerinte complexe de validare (Thrust-ul).
9. Button Bouncing si Software Debounce (Electronica / Informatica)
   Am constatat ca la eliberarea degetului de pe butonul fizic, sistemul inregistra eronat multiple “miscari fantoma”. Am identificat cauza hardware a fenomenului ca fiind micro-vibratia mecanica a lamelei de contact intern (bouncing). Am rezolvat problema la nivel de software prin implementarea unei secvente de debounce (blocarea temporara a executiei cu delay) pentru a filtra aceste oscilatii parazite.
10. Modulatia Latimii Pulsului (PWM) si Sinteza Aditiva (Electronica)
    Am tranzitat de la comutarea binara (digitalWrite) la generarea de semnale PWM (analogWrite) pentru a controla voltajul aplicat fiecarui canal al LED-ului RGB. Aceasta abordare a permis compensarea diferentelor fizice de eficienta luminoasa dintre materialele semiconductoare (unde verdele si albastrul sunt mult mai intense decat rosul), realizand o calibrare cromatica de precizie.
11. Generarea Procedurala de Sunet (Informatica / Acustica)
    Am identificat limitarile tehnice ale unui buzzer piezo pasiv (incapacitatea de a reda esantioane audio sau frecvente suprapuse) si am dezvoltat metode de sintetizare a anvelopelor sonore direct din cod. Utilizand iteratii procedurale (for) si moduland functia tone(), am variat matematic frecventele in timp real pentru a simula un clinchet magic cristalin (Flick), un efect Doppler de taiere a aerului (Thrust) si un fasait modulat pseudo-aleatoriu reprezentand vantul (Twist).

7.Implementare

Vezi main.cpp

```
#include <Arduino.h>
#include <Wire.h>
#define PIN_LED_R 4
#define PIN_LED_G 3
#define PIN_LED_B 5
#define PIN_BUTON 6
#define PIN_BUZZER 7
const int MPU_ADDR = 0x68;
int16_t AccelX, AccelY, AccelZ, GyroX, GyroY, GyroZ;
bool aFostApasat = false;
long minAX, maxAX, minAY, maxAY, minAZ, maxAZ;
long minGY, maxGY;
long minGX, maxGX;

//game info
int stadiuJoc = 0;
void afiseazaIntro();
int proceseazaPovestea(int stadiuCurent, int codMiscare);

void seteazaCuloare(int r, int g, int b) {
  analogWrite(PIN_LED_R, r);
  analogWrite(PIN_LED_G, g);
  analogWrite(PIN_LED_B, b);
}

void sunetMagieFlick() {
  for (int frecventa = 1500; frecventa < 3500; frecventa += 250) {
    tone(PIN_BUZZER, frecventa, 25);
    delay(30);
  }
  noTone(PIN_BUZZER);
}
void sunetSabieThrust() {
  for (int frecventa = 1200; frecventa > 300; frecventa -= 80) {
    tone(PIN_BUZZER, frecventa, 10);
    delay(12);
  }
  noTone(PIN_BUZZER);
}
void sunetVantTwist() {
  for (int i = 0; i < 15; i++) {
    int frecventaAleatorie = random(120, 240);
    tone(PIN_BUZZER, frecventaAleatorie, 20);
    delay(25);
  }
  noTone(PIN_BUZZER);
}
```

```
void sunetCercMoonSpell() {
  for (int i = 0; i < 4; i++) {
    tone(PIN_BUZZER, 600, 30);
    delay(40);
    tone(PIN_BUZZER, 1200, 30);
    delay(40);
  }
  noTone(PIN_BUZZER);
}

void setup() {
  Serial.begin(38400);
  Wire.begin();
  pinMode(PIN_BUTON, INPUT_PULLUP);
  pinMode(PIN_BUZZER, OUTPUT);
  pinMode(PIN_LED_R, OUTPUT);
  pinMode(PIN_LED_G, OUTPUT);
  pinMode(PIN_LED_B, OUTPUT);
  seteazaCuloare(0, 0, 0);
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);
  //filtru DLPF(Treapta 4=~20Hz)
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(0x1A);
  Wire.write(0x04);
  Wire.endTransmission(true);
  // accelometru la ±8g
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(0x1C);
  Wire.write(0x10);
  Wire.endTransmission(true);
  // giroscop: ±1000 grade/sec
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(0x1B);
  Wire.write(0x10);
  Wire.endTransmission(true);
  Serial.println("Bagheta magica v2.6!");
  Serial.println("Tine apasat butonul, executa miscarea, apoi
elibereaza-l.");
  afiseazaIntro();
}

void loop() {
  if (digitalRead(PIN_BUTON) == LOW) {
    if (aFostApasat == false) {
      aFostApasat = true;
      minAX = 32767;
      maxAX = -32768;
      minAY = 32767;
```

```
      maxAY = -32768;
      minAZ = 32767;
      maxAZ = -32768;
      minGY = 32767;
      maxGY = -32768;
      minGX = 32767;
      maxGX = -32768;
      //alb-calibrat: buton apasat
      seteazaCuloare(255, 100, 50);
    }
    //citim datele senzorului
    Wire.beginTransmission(MPU_ADDR);
    Wire.write(0x3B);
    Wire.endTransmission(false);
    Wire.requestFrom(MPU_ADDR, 14, true);
    AccelX = Wire.read() << 8 | Wire.read();
    AccelY = Wire.read() << 8 | Wire.read();
    AccelZ = Wire.read() << 8 | Wire.read();
    Wire.read(); Wire.read(); //useless temp
    GyroX = Wire.read() << 8 | Wire.read();
    GyroY = Wire.read() << 8 | Wire.read();
    Wire.read(); Wire.read();
    //caut extreme -while button press
    if (AccelX < minAX)
    {
      minAX = AccelX;
    }
    if (AccelX > maxAX)
    {
      maxAX = AccelX;
    }
    if (AccelY < minAY)
    {
      minAY = AccelY;
    }
    if (AccelY > maxAY)
    {
      maxAY = AccelY;
    }
    if (AccelZ < minAZ)
    {
      minAZ = AccelZ;
    }
    if (AccelZ > maxAZ)
    {
      maxAZ = AccelZ;
    }
    if (GyroX < minGX){
      minGX = GyroX;
    }
```

```
    if (GyroX > maxGX){
      maxGX = GyroX;
    }
    if (GyroY < minGY){
      minGY = GyroY;
    }
    if (GyroY > maxGY){
      maxGY = GyroY;
    }
    delay(5);
  }
  else {
    //buton eliberat
    if (aFostApasat == true) {
      //calc amplitudine:(Delta = forta + franarea din recul)
      long deltaX = maxAX - minAX;
      long deltaY = maxAY - minAY;
      long deltaZ = maxAZ - minAZ;
      long deltaGyroX = maxGX - minGX;
      long deltaGyroY = maxGY - minGY;
      // calc forta compusa pe x si y pt Thrust (miscare inainte)
      long deltaThrust = deltaY + deltaZ;

      int codMiscare = 0; // 0 = Invalida, 1=Flick, 2=Thrust, 3=Twist,
4=WindSword, 5=MoonSpell

      //DEBUG STUF !!!!!!!!!!
      // Serial.print("Debug -> Forta X: "); Serial.print(deltaX);
      // Serial.print(" | Forta Y: "); Serial.print(deltaY);
      // Serial.print(" | Forta Z: "); Serial.print(deltaZ);
      // Serial.print(" | Rotatie Y: "); Serial.println(deltaGyroY);

      // LOGICA DE DECIZIE EFECTE
      //
      //
      // SWORD KEY! - alb + dublu sunet
      if (deltaGyroY > 30000 && deltaThrust > 30000 && deltaY > 8000) {
          Serial.println("GEST COMPUS: KEY SWORD! (Impungere +
Rasucire)\n");
          codMiscare = 4;
          seteazaCuloare(255, 255, 255); // ALB pentru combo
          sunetSabieThrust();
          delay(5);
          sunetVantTwist();
      }
      // MOON SPELL - roz + mov
      else if (deltaGyroX > 15000 && deltaX > 10000 && deltaZ > 10000) {
          Serial.println("GEST COMPUS: MOON SPELL (Cerc)\n");
          codMiscare = 5;
          seteazaCuloare(255, 50, 50); //roz
          delay(150);
```

```
            seteazaCuloare(150, 0, 255); //mov
            sunetCercMoonSpell();
        }
        // TWIST- cian (verde 200 + albastru 150)
        else if (deltaGyroY > 30000) {
            Serial.println("GEST: KEY (Rasucire din incheietura)\n");
            codMiscare = 3;
            seteazaCuloare(0, 200, 150);
            sunetVantTwist();
        }
        //FLICK - mov (Rosu 255 + albastru redus la 80)
        else if (deltaX > 15000 && deltaX > (deltaZ * 1.3)) {
            Serial.println("GEST: FLICK (Scuturare laterala)\n");
            codMiscare = 1;
            seteazaCuloare(255, 0, 80);
            sunetMagieFlick();
        }
        //THRUST-galben (Rosu 255 + verde redus la 120)
        else if (deltaThrust > 13000) {
            if (deltaX < 12000) {
                Serial.println("GEST: SWORD (Impungere in fata)\n");
                codMiscare = 2;
                seteazaCuloare(255, 120, 0);
                sunetSabieThrust();
            } else {
                Serial.println(" MISCARE RESPINSA: Prea haotica\n");
                seteazaCuloare(255, 0, 0); // Rosu curat
            }
        }
        else {
            Serial.println("MISCARE IGNORATA: Forta prea mica.\n");
            seteazaCuloare(0, 0, 0);
        }
        stadiuJoc = proceseazaPovestea(stadiuJoc, codMiscare);

        aFostApasat = false;
        //Lasam culoarea vizibila inca putin, apoi stingem
        delay(200);
        seteazaCuloare(0, 0, 0);
        delay(100); //anti-bouncing
      }
    }
}
```

Vezi poveste.cpp

```
#include <Arduino.h>

//Lista coduri - 1=FLICK, 2=THRUST, 3=TWIST, 4=WIND SWORD, 5=MOON SPELL
```

```
void afiseazaIntro() {

Serial.println(F("\n-----------------------------------------------------")
);
    Serial.println(F("BINE AI VENIT!"));
    Serial.println(F("HAI SA INCEPEM AVENTURA..."));
    Serial.println(F(""));
    Serial.println(F("POVESTE:"));
    Serial.println(F("Esti intr-o padure intunecata. Nu stii cum ai ajuns
aici sau cine esti. Iti dai seama ca strangi puternic ceva in mana... O
BAGHETA MAGICA!"));
    Serial.println(F(""));
    Serial.println(F("Nu ai timp sa o admiri. Ceva in coltul privirii se
misca! Tufirusile din jurul tau par sa-si indrepte ramurile ghemuite spre
tine."));
    Serial.println(F(""));
    Serial.println(F("Maracini ca niste unghii colturoase se apropie de
fata ta. Instinctul ia controlul si mana ti se misca. Da la o parte
ramurile spinoase!"));
    Serial.println(F(""));
    Serial.println(F(""));
    Serial.println(F("TUTORIAL: Pentru a folosi bageta, tine apasat butonul
in timpul intregii miscari."));

Serial.println(F("-----------------------------------------------\n"));
}

int proceseazaPovestea(int stadiuCurent, int codMiscare) {
    //capitole (un capitol, o miscare diferita)
    switch (stadiuCurent) {
        case 0: // FLICK efectuat
            if (codMiscare == 1) {
                Serial.println(F("\n---GEST EXECUTAT PERFECT!---"));
                Serial.println(F("CAPITOLUL 1:"));
                Serial.println(F("Ai facut prima vraja! Un arc mov reteaza
toate radacinile! Palcul spinos si intunecat in care te aflai se lumineaza
si se deschide. Raza ta vizuala se largeste."));
                Serial.println(F(""));
                Serial.println(F("Dar si RAZA LUI VIZUALA SE LARGESTE! Un
monstru te-a observat!"));
                Serial.println(F(""));
                Serial.println(F("Repede! Impunge-l cu bagheta!"));
                Serial.println(F(""));
                Serial.println(F("----------------------------\n"));
                return 1; //cod pt urmatorul chapter
            } else if (codMiscare == 0){
                Serial.println(F("HINT: Miscarile sunt simple, nu mai mult
de o secunda. Cum ai da la o parte ramuri din fata ta?"));
                return 0; // Ramanem la stadiul 0
            } else {
                Serial.println(F("Easy Cowboy! Nu o lua inainte... Hai sa
```

```
incepem cu inceputul... (ce ai facut tu e o miscare mai avansata)"));
                return 0; // Ramanem la stadiul 0
            }
            break;

        case 1: //THRUST efectuat
            if (codMiscare == 2) {
                Serial.println(F("\n---GEST EXECUTAT PERFECT!---"));
                Serial.println(F("CAPITOLUL 2:"));
                Serial.println(F("A mers! Nu chiar cum te asteptai? Bagheta
s-a transformat intr-o sabie pentru o secunda si a dezintegrat monstrul la
impact!"));
                Serial.println(F(""));
                Serial.println(F("In urma lui a ramas un cufar cu o aura
misterioasa. In mijloc are o gaura de cheie... Hm.. Oare nu ai putea
sa...?"));
                Serial.println(F(""));
                Serial.println(F("------------------------------\n"));
                return 2; // Avansam la stadiul 2
            } else if (codMiscare == 0){
                Serial.println(F("HINT: Miscarile sunt simple, nu mai mult
de o secunda. \"Repede! Impunge-l cu bagheta!\""));
                return 1; // Ramanem la stadiul 1
            } else {
                Serial.println(F("Asta e alt spell..."));
                return 1; // Ramanem la stadiul 1
            }
            break;

        case 2: //Twist efectuat
            if (codMiscare == 3) {
                Serial.println(F("\n---GEST EXECUTAT PERFECT!---"));
                Serial.println(F("CAPITOLUL 3:"));
                Serial.println(F("Exemplu: Ai deschis cufarul! Exact cum te
asteptai, bagheta s-a transformat intr-o cheie!"));
                Serial.println(F(""));
                Serial.println(F("Dar la ce nu te asteptai e sa fi trasa
inauntrul cufarului! Of.. era o capcana.."));
                Serial.println(F(""));
                Serial.println(F("Acum te afli intr-o incapere uriasa cu
pereti inalti si arhitectura gotica. Aproape ca un castel in ruine."));
                Serial.println(F(""));
                Serial.println(F("In fata ta e un monstru de secrete.
Corpul lui e ca o cenusa vascoasa, imposibil de ranit. Totusi, in centrul
pieptului are un lacat de fier. E imposibil de spart cu o simpla cheie,
trebuie sa-i dai si forta."));
                Serial.println(F(""));
                Serial.println(F("------------------------------\n"));
                return 3;
            } else if (codMiscare == 0){
```

```
                Serial.println(F("HINT: Miscarile sunt simple, nu mai mult
de o secunda.\n \"Bagheta s-a transformat intr-o sabie pentru o
secunda(...). Oare nu ai putea sa...?\""));
                return 2;
            }else {
                Serial.println(F("Asta e alt spell..."));
                return 2; // Ramanem la stadiul 2
            }
            break;

        case 3: //SWORD KEY facut
            if (codMiscare == 4) {
                Serial.println(F("\n--- GEST EXECUTAT PERFECT! ---"));
                Serial.println(F("CAPITOLUL 4:"));
                Serial.println(F("Ti-ai luat avant si ai facut cea mai
letala cheie din istoria magiei! Jumatate cheie si jumatate pumnal, nu avea
bietul lacat nicio sansa..."));
                Serial.println(F(""));
                Serial.println(F("Acum toate secretele monstrului au scapat
ca licuricii. Zboara haotic si se indreapta spre geamurile sparte ale
incaperii, fiecare intr-o directie diferita."));
                Serial.println(F(""));
                Serial.println(F("Prinde-i repede! Ce forma ii poate
tine?"));
                Serial.println(F(""));
                Serial.println(F("------------------------------\n"));
                return 4;
            } else if (codMiscare == 0){
                Serial.println(F("\nHINT: Ai putea incerca sa combini doua
miscari cunoscute.."));
                return 3;
            }else {
                Serial.println(F("Asta e alt spell..."));
                return 3; // Ramanem la stadiul 2
            }
            break;

        case 4: //MOON SPELL facut
            if (codMiscare == 5) {

Serial.println(F("\n-------------------------------------------"));
                Serial.println(F("VICTORIE!"));
                Serial.println(F("Bagheta s-a miscat sub indrumarea lunii.
Ai facut un cerc, iar licuricii de memorie s-au adunat in palma ta stanga,
intr-o bila argintie"));
                Serial.println(F(""));
                Serial.println(F("Ia mana de pe bagheta. Acum iti
amintesti... Esti la PM fair si ai intrat in examen!!! YAAYY"));
                Serial.println(F(""));

                Serial.println(F(">>> Joaca din nou facand orice miscare de
```

```
FLICK."));

Serial.println(F("-----------------------------------------\n"));
                return 5; // Stadiul final, asteapta restart
            } else if (codMiscare == 0){
                Serial.println(F("\nHINT: Luna pare sa-ti sopteasca ceva
printre cioburile de geam.."));
                return 4;
            }else {
                Serial.println(F("Asta e alt spell..."));
                return 4;
            }
            break;

        case 5: // Restart Joc
            if (codMiscare == 1) {
                Serial.println(F("\n\n>>> RESTARTING GAME...\n\n"));
                afiseazaIntro();
                return 0;
            } else {
                return 5;
            }
            break;

        default:
            return stadiuCurent;
    }
}
```

8. Laboratoarele Folosite

Protocolul I2C (Wire.h): Utilizarea I2C a fost obligatorie pentru a permite comunicatia sincrona de mare viteza intre microcontroller-ul AVR si registrul intern al senzorului MPU6050. Fara I2C, extragerea celor 6 axe de date (accelerometru 3D si giroscop 3D) in timp real (la ~200Hz) ar fi fost imposibila din cauza limitarilor de pini fizici.

Interfata UART (Serial): Implementarea comunicatiei seriale asincrone a fost cruciala atat in faza de cercetare (pentru transmiterea datelor brute catre plotter-e externe in vederea extragerii pragurilor matematice), cat si in faza finala, servind ca interfata de debug si consola principala de output a deciziilor sistemului.

Pinii GPIO (Input/Output Digital)(citirea starii logice a butonului cu rezistenta interna de pull-up) si comanda bruta a modulului piezo-acustic (generarea undelor patrate prin functia tone()).

Modulatia PWM (analogWrite)(ex: ajustarea verdelui si albastrului pentru a nu suprima rosul).

Rezultate Obţinute

https://youtu.be/O2fivbEIWfQ

Se observa in video in ordinea asta:

1. Flick Spell: Miscare dreapta/ stanga bagheta
2. Sword (Thrust) Spell: Miscare bagheta fata
3. Key (Twist) Spell: Rotire Bagheta
4. Sword Key (Thrust + Twist) Spell: Miscare compusa din ambele actiuni anterioare
5. Moon (Circle) Spell: Cerc

Actiunile se fac in ordinea in care se cere in joc, de aia se observa progresul (apare text poveste dupa fiecare actiune corecta in video). Ultimele 2 actiuni din video sunt:

1. Flick → replay game
2. Twist → actiunea gresita pt Capitolul 1: apare un scurt mesaj esec.

Cum functioneaza sistemul si jocul:

Detectia Euristica a Miscarii: Sistemul preia date brute pe 6 axe (accelerometru + giroscop) doar cat timp butonul baghetei este mentinut apasat. Folosind un algoritm propriu bazat pe diferente absolute (“Metoda Delta”), bagheta face distinctia intre 5 gesturi (Flick, Thrust(Sword Spell), Twist(Key Spell), Twist+ Thrust (Key-Sword Spell) si Circle (Moon Spell)) fara a folosi putere de calcul mare sau Machine Learning.

Mecanica Jocului (State Machine): Software-ul functioneaza ca o “masina de stari” secventiala. Sistemul afiseaza in consola o situatie si cere utilizatorului sa execute o miscare magica specifica pentru a trece mai departe.

Progresie si Feedback: * Daca jucatorul executa gestul corect, bagheta confirma succesul declansand o combinatie unica de lumini pe LED-ul RGB si un sunet procedural din buzzer, iar povestea avanseaza la urmatorul nivel (Fiecare spell are lumina si sunetul propriu. Key Sword are sunetul combinat dintre Sword Spell si Key Spell)

Daca miscarea este gresita sau confuza, sistemul returneaza vizual si auditiv o eroare, iar utilizatorul trebuie sa repete actiunea.

Optimizare Hardware: Jocul ruleaza stabil fara sa blocheze microcontroller-ul. Limitarea stricta de 2KB a memoriei RAM a fost depasita prin mutarea intregului text al jocului in memoria Flash a procesorului.

Concluzii

Fun proiect. :D

Download

pm_magiccast.zip

Bibliografie/Resurse

Resurse Hardware:

- https://ww1.microchip.com/downloads/en/DeviceDoc/50002659A.pdf (datasheet ATmega324P Xplained Mini)
- https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42743-ATmega324P_Datasheet.pdf (datasheet ATmega324P → pentru diagrame electrice)
- https://mysii.gorriens.net/images/arduino/capteurs/gy-521_mpu-6050_3-axis_gyroscope_and_acceleration_sensor_en.pdf(datasheet accelerometru GY-521mpu-6050)
- https://www.snapeda.com/parts/GY-521/GODREAM%20FORDREAM%20CO.,%20LIMITED/view-part/?welcome=home&ref=search&t=GY-521&ab_test_case=b (biblioteca piesa accelerometru pt. Fusion)

Resurse Software:

- https://www.arduino.cc/reference/en/ (Documentatia oficiala a limbajului Arduino si a functiilor de baza utilizate: analogWrite, tone, millis, etc.)
- https://www.arduino.cc/en/Reference/Wire (Referinta pentru libraria Wire.h, utilizata pentru comunicatia I2C)
- https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf (Harta registrelor MPU6050 - vitala pentru scrierea la nivel de bit a setarilor de filtru DLPF si scalare a accelerometrului/giroscopului)
- https://platformio.org/ (PlatformIO - ecosistemul de dezvoltare instalat in VSCode, utilizat pentru compilare, incarcare cod si monitorizarea consumului de memorie)
- https://marketplace.visualstudio.com/items?itemName=alexnesnes.teleplot (Teleplot - extensia de VSCode folosita intensiv pentru plotarea grafica a valorilor MPU6050 in timp real si extragerea pragurilor pentru “Metoda Delta”)
- https://www.arduino.cc/reference/en/language/variables/utilities/progmem/ (Documentatia pentru utilizarea memoriei Flash si a macro-ului F() pentru salvarea memoriei SRAM la afisarea textelor din consola)