Automatic Cat Toy

Nume: Popescu Ioana-Denisa

Grupa: 336CA

Introducere

Considerate animale de companie crepusculare, majoritatea pisicilor domestice își manifestă activitatea noaptea, având un nivel ridicat de energie și dorință de joacă. Din păcate, factori precum programul încărcat și dorința naturală de odihnă îi împiedică pe proprietari să interacționeze cu felinele, care tânjesc, conform studiilor, după atenție umană.

Proiectul **Automatic Cat Toy** își propune să rezolve această problemă, venind ca o punte de mijloc care să împace ambele părți: felina se joacă fericită, iar proprietarul are o odihnă liniștită.

Descriere generală

Schema Bloc

Proiectul constă în implementarea unei jucării automate pentru pisici, componenta principală pe care se bazează fiind Arduino UNO ATmega328P. Vor exista, în principal, 2 moduri de funcționare, care vor putea fi selectate prin intermediul unui buton:

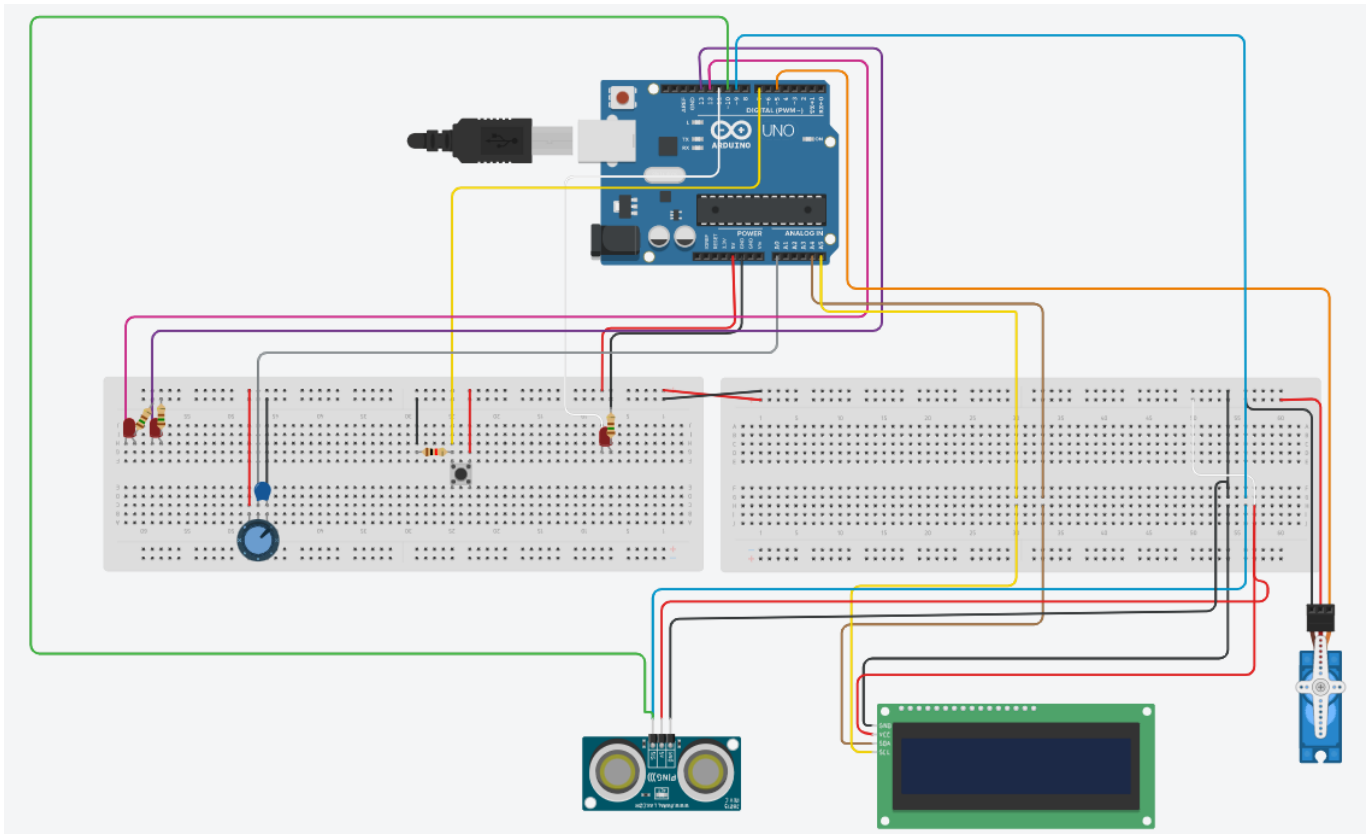
- **Modul activ** - jucăria se mișcă continuu la diferite intensități și unghiuri, care pot fi reglate prin intermediul unui potențiometru. Mișcarea se face printr-un ServoMotor, stabilizat si legat de un ansamblu special construit. În acest mod, se va aprinde un led de o anumită culoare (roșu/verde) în funcție de unghi și se va afișa pe ecran o mică animație specifică.
- **Modul senzor** - se aprinde un led galben, iar inițial, jucăria stă nemișcată. Mișcarea se va declanșa prin intermediul unui senzor, încadrat într-un suport, care va măsura distanța minimă față de pisică, la un nivel inferior, la care aceasta are acces. În momentul în care pisica este în zona senzorului, jucăria face o scurtă mișcare, apoi revine în starea inițială până la o nouă înregistrare pe senzor. Pe LCD se va afișa un mesaj care conține informații despre distanța la care se află pisica.

Hardware Design

Listă de piese

- Arduino UNO - ATmega328P
- Ecran LCD
- Modul interfață I2C LCD
- ServoMotor SG90
- Senzor Ultrasonic HC-SR04
- Potențiometru
- 3 Led-uri
- 3 rezistențe de 150 Ω
- 1 rezistență de 1K Ω
- 1 buton
- Legătură cu un calculator pentru interfața serială
- Fire, 2 Breadboard-uri
- Condensator 2.2 nF

Schemă Hardware

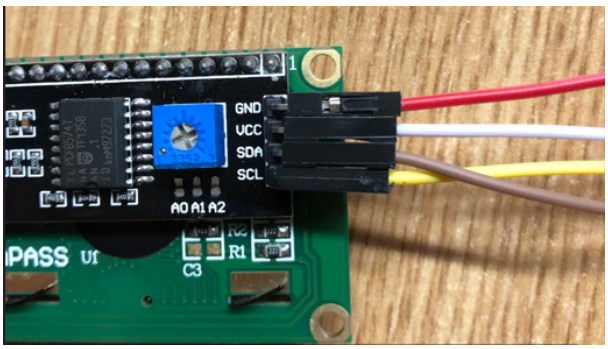


Conectare componente

Modul interfață I2C LCD

Folosit atât pentru a reduce numărul pinilor necesari conectării Arduino, cât și pentru manevrarea cu ușurință a contrastului de pe LCD cu ajutorul potențiometrului incorporat.

- GND → GND
- VCC → 5V
- SDA (Serial Data Pin) → A4
- SCL (Clock Pin) → A5



ServoMotor SG90

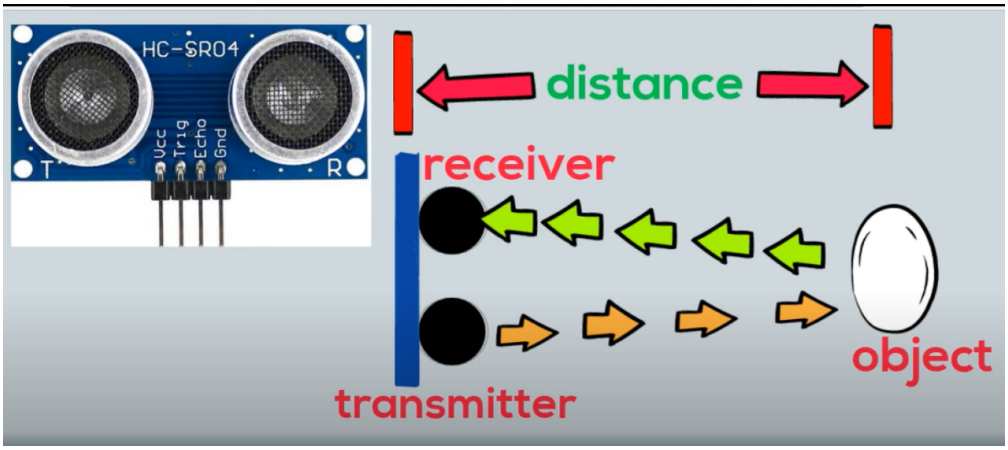
Necesar mișcării jucăriei, atât la o poziție predefinită, cât și întâmplătoare, folosind PWM.

- GND → GND
- VCC → 5V
- PWM → PD5

Senzor Ultrasonic HC-SR04

Măsoară distanța până la obiectul detectat, înmulțind timpul dintre transmiterea semnalului sender si primirea celui reciever (reflectat) cu viteza luminii.

- GND → GND
- VCC → 5V
- Trig (trimite pulsul) → PD9
- Echo (identifică reflexia de la obiect) → PD10



Potențiometru

Utilizat pentru a mișca servomotorul cu un anumit unghi, prin control variabil asupra semnalelor. Aici s-a folosit un convertor analog-digital, iar pinii de GND și Wiper au fost puși ulterior în paralel cu un condensator de 2.2 nF pentru stabilitate (valorile se aflau inițial într-o fluctuație continuă și era dificil de aproximat un rezultat).

În funcție de valoarea sa, se aprind ledurile succesiv: cel roșu pentru valori mai mari de 500, altfel, cel verde.

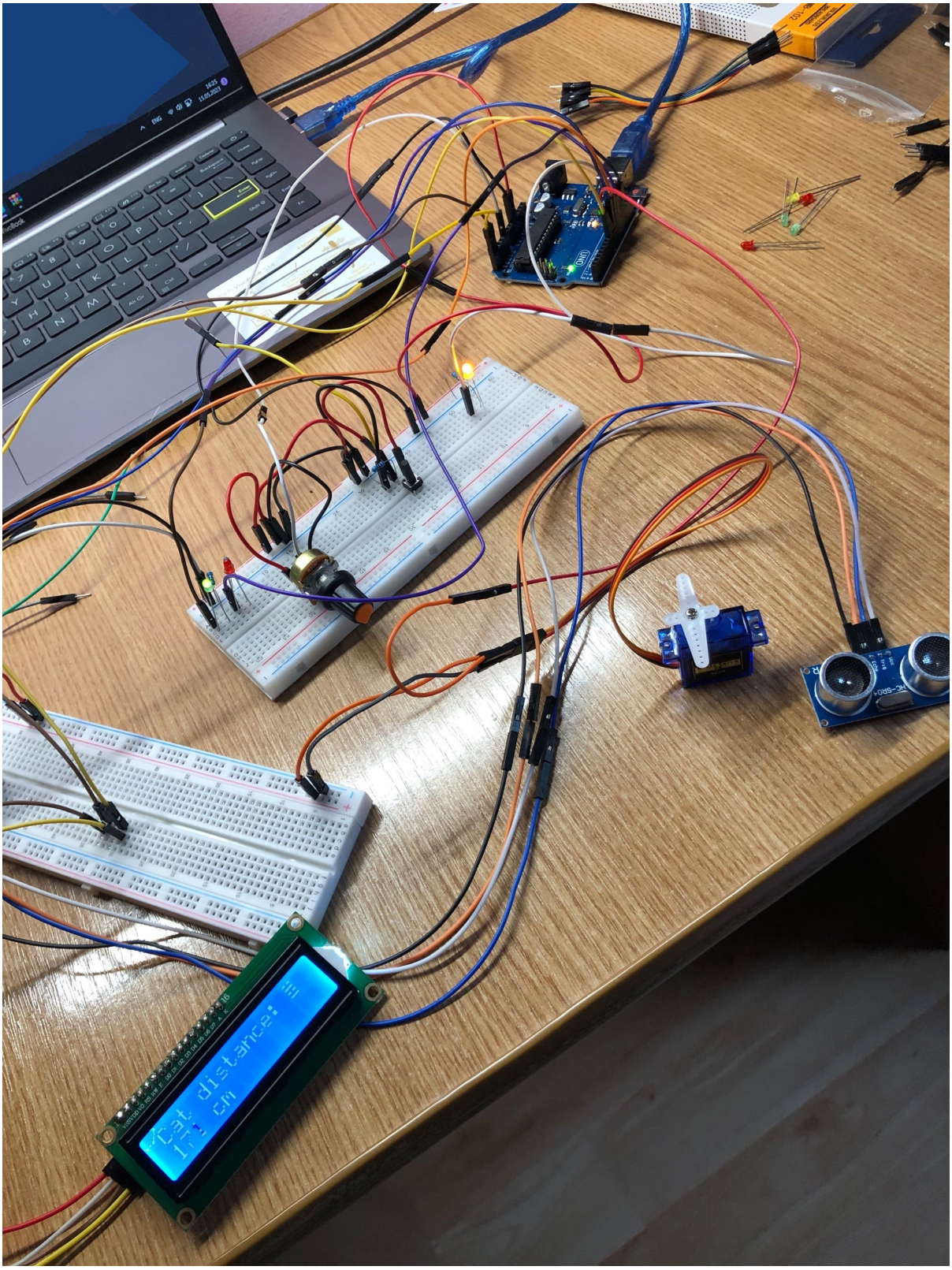
- T1 → GND
- Wiper → A0
- T2 → VCC

Buton

La apăsare, se schimbă modurile de funcționare și se aprinde ledul galben. Determinarea stării butonului se face cu o întrerupere de tip Pin Change Interrupt.

- Terminal 1b → D7
- Terminal 2b → VCC

Circuit Fizic (variantă inițială)



Rezultatele simulării

În această etapă am conectat componentele Hardware și am scris si o parte din cod, bazându-mă strict pe funcții Arduino. Rezultatul a fost unul așteptat, funcționalitățile de bază mergeau, însă logica programului nu era încă implementată și optimizată, iar designul era încă la nivel teoretic. Ulterior, circuitul a fost puțin modificat deoarece am constatat că pinii pentru servomotor nu erau potriviți, am adăugat un condensator și am schimbat puțin poziționarea pentru a elimina firele redundante.

Software Design

În ceea ce privește partea de software, am folosit Arduino IDE cu următoarele biblioteci:

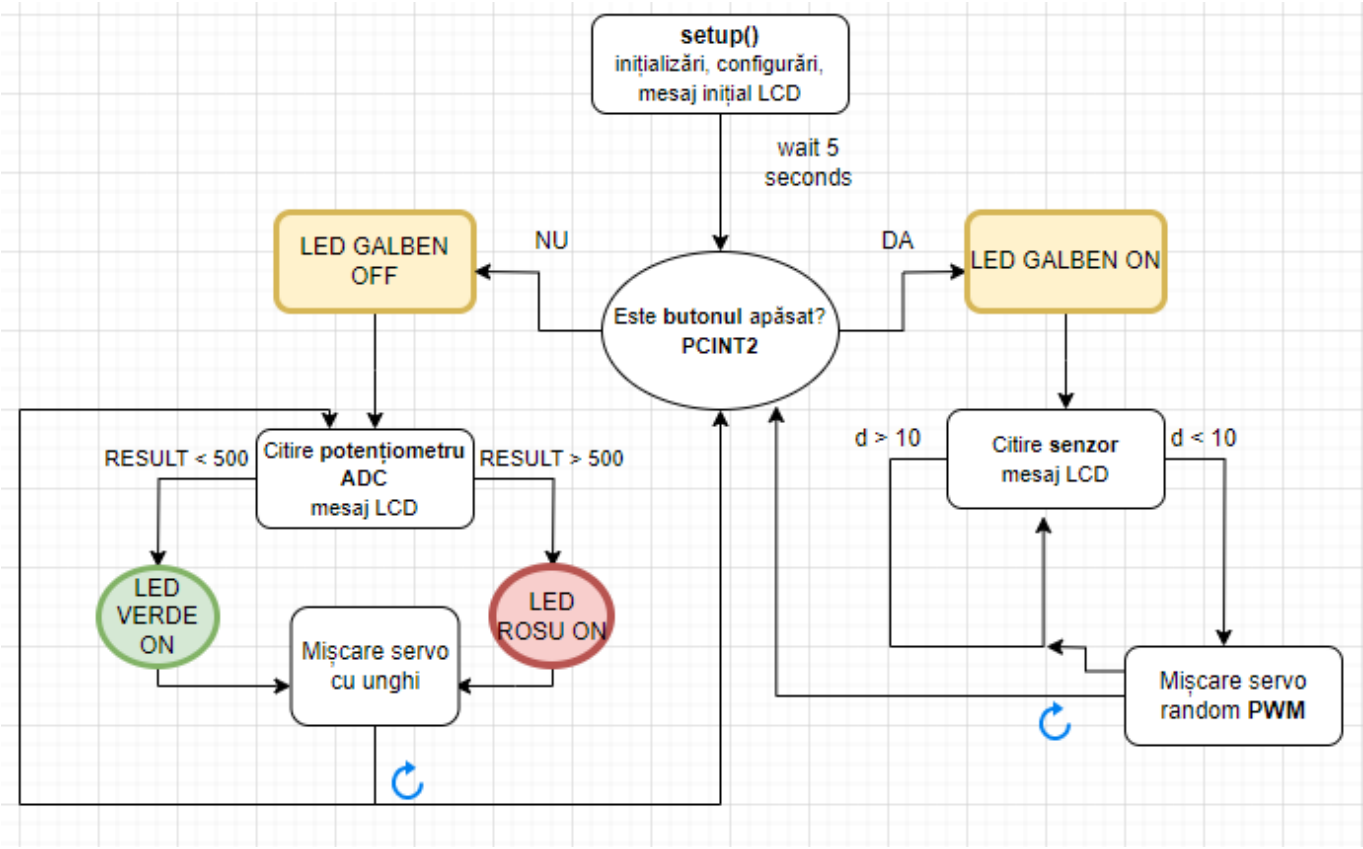
- **Servo.h** pentru controlul servomotor-ului
- **LiquidCrystal_I2C.h** pentru LCD
- **Wire.h** pentru scrierea pe LCD

Ca idee generală, componentele interacționează prin variabile partajate, declarate **volatile** sau **static**. Pe lângă funcțiile clasice Arduino utilizate, cele mai interesante concepte software de care m-am folosit au fost:

1. **Pin Change Interrupt** (Timer 2)
2. **PWM** (Compare Match Interrupt, Timer 0)
3. **ADC**

Workflow

Logica programului este ilustrată în schema de mai jos. Modul de funcționare default este modul 1, cu mișcarea în funcție de potențiometru.



Întrerupere Pin Change pe Timer 2

Pentru întreruperea de tip Pin Change de la buton, am folosit Timer-ul 2. Aici aflăm practic modul de

funcționare. Inițial, am făcut următoarele configurări:

```
PCICR |= B00000100;   // set Timer 2
PCMSK2 |= B10000000;  // PCINT23 = PD7
```

Așa arată rutina de întrerupere. Am dezactivat Timer-ul 0 în timpul apăsării pentru a evita interferențe.

```
ISR(PCINT2_vect) {

 int buttonStatus = digitalRead(button);  // Read the current button status

 if (buttonStatus && !buttonPressed) {   // Button was just pressed

  digitalWrite(led, !digitalRead(led));  // Toggle the yellow LED status
  buttonPressed = true;
  TIMSK0 &= ~_BV(OCIE0A);                // Disable Timer0 interrupt during
button press

 } else if (!buttonStatus && buttonPressed) {  // Button was just released
   buttonPressed = false;
   TIMSK0 |= _BV(OCIE0A);                    // Enable Timer0 interrupt
during button press
 }
}
```

Întrerupere Compare Match pentru Timer 0 (PWM)

Ideea aici a fost să mișc servomotorul în poziții cât mai random, la un interval regulat, atunci când senzorul detectează mișcarea unui obiect apropiat. În fucția setup(), am făcut următoarele inițializări:

```
TCCR0A = _BV(COM0B1) | _BV(WGM00) | _BV(WGM01);  // Non-Inverting Fast PWM
Mode, Fast PWM, 0-255, resets to 0
TCCR0B = _BV(WGM02) | _BV(CS02) | _BV(CS00);     // Waveform Generation Mode,
Prescaler 1024
OCR0A = 31;                                       // Set TOP value
OCR0B = 15;                                       // Set initial value
TIMSK0 |= _BV(OCIE0A);                            // Enable Timer 0 Compare
Match
sei();                                            // Enable interrupts
```

Calcule internal counter frequency

f_int = 16,000,000 / (1024 * (255 + 1)) = 16,000,000 / 262144 = 61.0352 Hz

Rutina de întrerupere, modul 2 de funcționare. În funcție de valoarea counter-ului Timer, calculez o nouă poziție de deplasare și un nou unghi pentru mișcarea servomotorului.

```
ISR(TIMER0_COMPA_vect) {
 Time++;                     // Global volatile variable
 if (Time == 20) {
```

```
   // Move servo
   if (forward) {
     pos += 5;              // Increment position
     if (pos >= 180)
       forward = false;     // Reached the maximum position, change
direction
   } else {
     pos -= 5;              // Decrement position
     if (pos <= 0)
       forward = true;      // Reached the starting position, change
direction
   }
   move = true;
   Time = 0;                // Reset timer
 }
}
```

ADC

După stabilizarea potențiometrului cu ajutorul condensatorului, am configurat citirea valorii în felul următor:

```
ADMUX |= B01000000;      // 5 V
ADCSRA |= B01000111;     // Prescaler 128 and Enable
ADCSRA |= (1 << ADSC);   // ADC Start Conversion
while (!(ADCSRA & (1 << ADIF))); // Wait
uint16_t result = ADC;
```

Apoi transform valoarea citită într-un unghi pentru servomotor, folosind funcția map. În cod se observă utilizarea unei formule pentru aproximarea rezultatului, întrucât potențiometrul, chiar și cu acel condensator în paralel, avea momente de instabilitate.

```
static uint16_t smoothedResult = result;
const float smoothingFactor = 0.5;        // Smoothing factor - for more
control over servo
```

```
smoothedResult = smoothingFactor * smoothedResult + (1 - smoothingFactor) *
result;
int servoPosition = map(smoothedResult, 0, 1023, 0, 180); // map value to
get angle
myServo.write(servoPosition); // move servo
```

Restul funcțiilor, elementele de sincronizare și dependențele dintre variabile sunt explicate mai detaliat în README-ul și codul din arhivă:

- drawAnimations → Afișează pe LCD titlul și animația inițială cu 5 pisici care dispar succesiv până la începerea jocului.
- updateLCD → Se scrie un mesaj pe LCD, în modul de funcționare 2, la un interval de timp, care să nu fie prea rapid (probleme cu PWM).
- moveServo() → Funcție care miscă efectiv servomotorul.

- nonBlockingServoControl() → Aceasta este o funcție care să miște servomotorul în modul de funcționare 2, la o poziție random calculată în rutina de întrerupere.

Rezultate Obținute

Rezultatul este o **jucărie funcțională**, cu un **design atractiv**, care îi oferă mobilitate, foarte interesantă pentru pisici, așa cum se poate observa și din demo-ul de mai sus!

Consider că rezultatul final depășește simularea inițială, cu funcții Arduino, care bifa criteriile de funcționalitate, însă la capitolul feature-uri și fiabilitate lăsa de dorit.

Concluzii

A fost un proiect foarte interesant, primul de până acum care implică componente fizice și lucru manual pentru design. Am lucrat cu mare drag la el, mai ales având în vedere scopul său și m-am bucurat foarte tare să îmi văd ideea pusă în practică după atâtea săptămâni. Bineînțeles, au existat și o serie de situații și evenimente neprevăzute, din care am putut trage o serie de concluzii:

- La un astfel de proiect, trebuie să ai foarte clar în minte ce vrei să faci și mai ales **de ce ai nevoie** exact. S-a întâmplat să am nevoie ulterior primei comenzi de rezistențe, alt tip de buton, condensator, etc.
- Există posibilitatea să nu primești componentele pe care le-ai comandat, evident o mare problemă dacă ai nevoie de ceva specific.
- **Lucrul cu întreruperi și AVR** este dificil, îți consumă timp, dar te face să înțelegi și să descoperi ce se află în spatele funcțiilor Arduino și poți pune în aplicare ce vezi în DataSheet.
- Să implementezi un proiect cu componente fizice este mult mai interesant ca pe TinkerCad.
- Multe biblioteci și funcții folosesc deja în spate Timer 1, deci trebuie atenție mare.
- Biblioteca LiquidCrystal nu este chiar atât de ușor de manipulat și poate cauza probleme în combinație cu rutina de întreruperi.
- **Servomotorul consumă foarte mult** curent/resurse și generează **electrical noise** care poate interfera cu modulul I2C de pe LED.
- Ar fi fost de preferat, în cazul proiectului meu, **să alimentez LCD-ul la altă sursă**, sau tot circuitul la o sursă peste 5V.
- **Potențiometrul** are fluctuații foarte mari și este foarte delicat, ajută să pui în paralel cu pinii săi un **condensator**.

Download

Arhiva cu codul sursă și README cu detalii despre implementare:

automatic_cat_toy_v2.zip

Jurnal

- **20 Aprilie** - Alegere temă proiect
- **26 Aprilie** - Confirmare temă, discuție la laborator
- **5 Mai** - Documentație Inițială
- **6 Mai** - Comandă piese
- **9-13 Mai** - Livrări
- **14 Mai** - Magazin fizic - ServoMotor, Led-uri, Rezistențe
- **21 Mai** - Hardware și funcționalități de bază
- **22 Mai** - Implementare logică și AVR
- **24 Mai** - Implementare buton întrerupere Pin Change
- **26 Mai** - Întrerupere PWM, schimbări sumare hardware
- **27-28 Mai** - Design și Finalizare cod
- **29 Mai** - Finalizare Design, Introducere condensator, testare

Bibliografie/Resurse

[1]: https://www.youtube.com/watch?v=GjWr48w6o2Q

[2]: https://www.youtube.com/watch?v=vf2lW4LkmMQ

[3]: https://ocw.cs.pub.ro/courses/_media/pm/atmel-7810-automotive-microcontrollers-atmega328p_datasheet.pdf

[4]: https://www.hackster.io/jacoslabbert99/arduino-lcd-icons-custom-characters-548f38

[5]: https://www.youtube.com/watch?v=aQy3DGSIGm4

[6]: https://www.youtube.com/watch?v=Uv9UeYUsA8A&t=192s

[7]: https://lastminuteengineers.com/i2c-lcd-arduino-tutorial/

[8]: https://forum.arduino.cc/t/rotary-encoder-interrupt-routine-interfering-with-servo-h/377538/5

[9]: https://forum.arduino.cc/t/lcd-over-i2c-not-working-with-timer2/523211/5

[10]: https://thewanderingengineer.com/2014/08/11/arduino-pin-change-interrupts/

[11]: https://chat.openai.com/

[12]: https://youtube.com/shorts/_sKWP6fl-NU?feature=share

Export to PDF

From:
http://ocw.cs.pub.ro/courses/ - **CS Open CourseWare**

Permanent link:
http://ocw.cs.pub.ro/courses/pm/prj2023/apredescu/automatic_cat_toy

Last update: **2023/05/29 22:42**