# Ruletă electronică

## Introducere

Aceasta este o ruletă electronică menită să simuleze o ruletă de cazino. Are tot ce ar avea o ruletă normală de cazino, cu excepția roții care se rotește, și a bilei – motiv pentru care ruleta aceasta este electronică. Include o modalitate de a selecta pe ce pariezi, un acceptor și un dispensor de monezi, ”ruleta” și ”bila” (reprezentată printr-un strip circular de led-uri prin care se va simula traseul bilei cum se rotește), o modalitate de a vedea rezultatul pariului (LCD + difuzor), și o manetă minusculă pentru a lansa bila. Jocul este făcut pentru a fi jucat de maxim o persoană odată.

Scopul ruletei este de a putea juca acest joc și fără acțiunea mecanică în sine, și asta este și ideea de la care am pornit. Cred că este utilă deoarece acest set-up permite jucarea acestui joc fără să fie nevoie de un dealer care să se ocupe de el (pe scurt, se poate automatiza).

## Descriere generală

Componente pentru functionalitatile principale: sursă de alimentare, acceptor de monezi, dispensor de monezi, difuzor, microcontroller, strip de Smart LEDs, senzor de îndoire, sticlă cu touch, LCD.

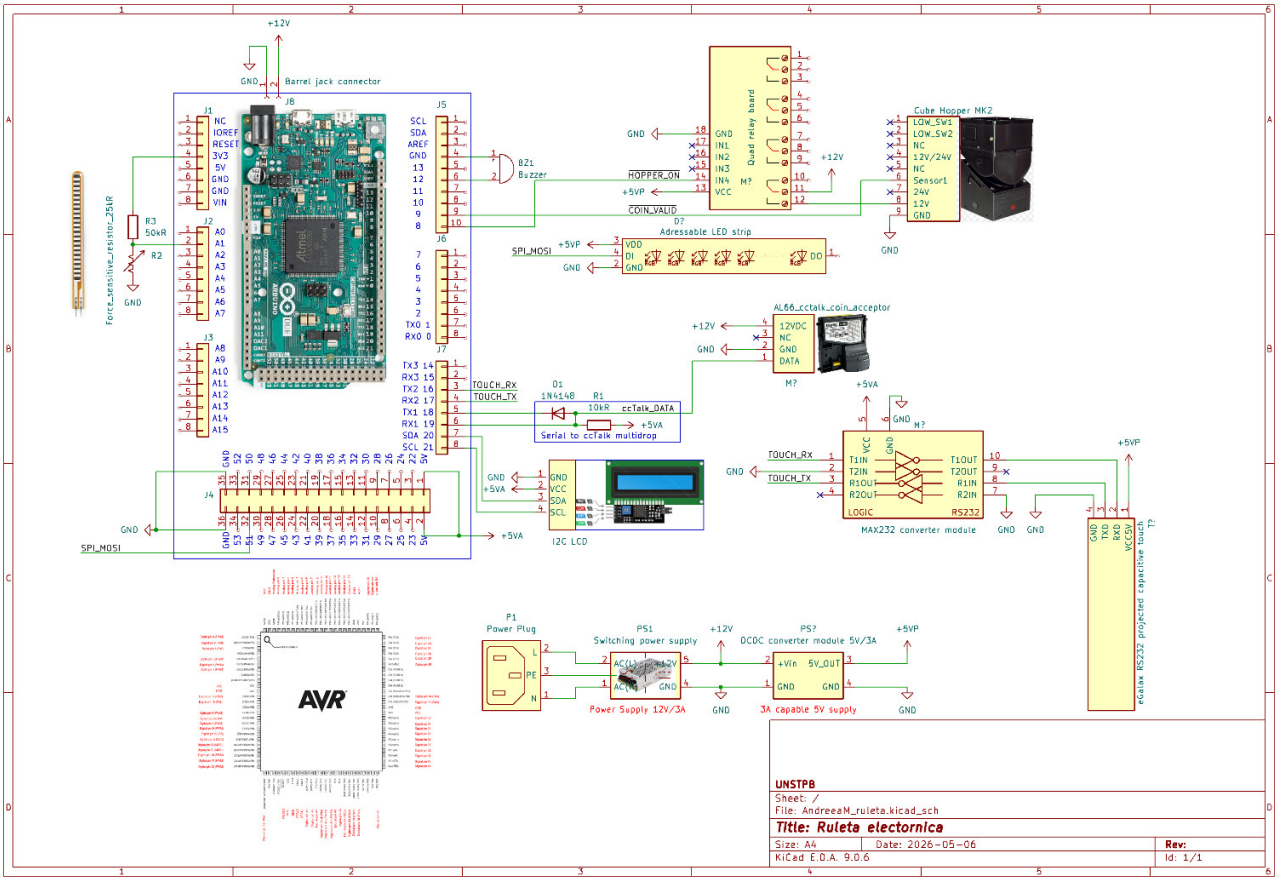
Acceptorul, dispensorul și microcontrollerul sunt conectate fiecare la sursa de alimentare.

Flow al interacțiunii componentelor: touch glass-ul (folosit pentru alegerea pariurilor) comunică prin UART cu microcontrollerul. Senzorul de îndoire (maneta de acțiune a “bilei”) transmite datele prin ADC. Microcontrollerul controlează ledurile folosind SPI (pentru a imita mișcarea bilei), și comunică cu difuzorul (PWM) și LCD-ul (I2C) pentru a arăta feedback-ul corespunzător (rezultatele jocului și pariului). Pentru tranzacțiile cu monezi, unitatea de control comunică cu acceptorul (UART) și dispensorul (GPIO) de monezi.

Schema bloc:

## Hardware Design

Schema electrică:




Listă a pieselor:

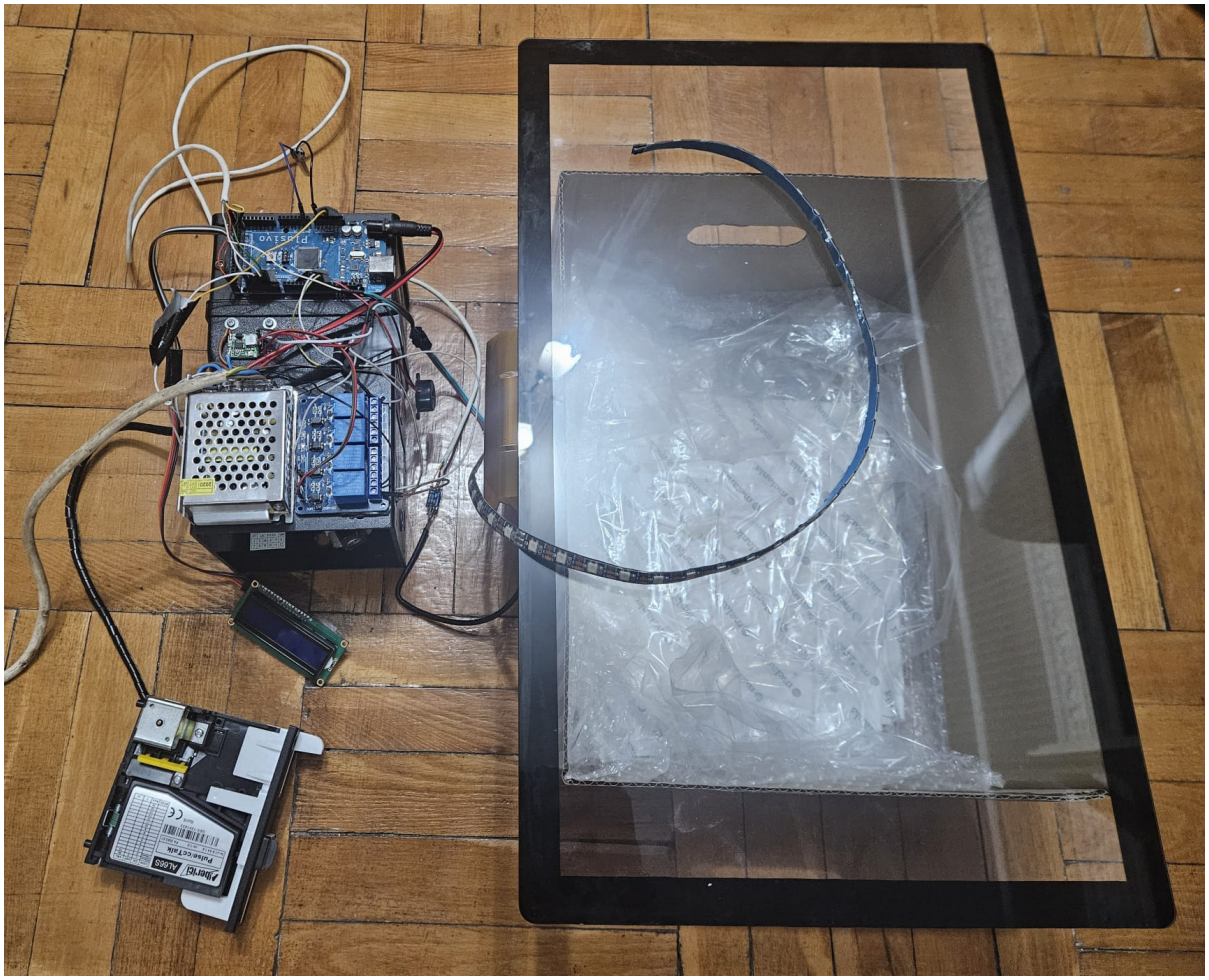
| Nr. Crt. | Piesă | Descriere |
|---|---|---|
| 1 | Alberici AL66S | acceptor de monezi |
| 2 | Cube Hopper MKII | dispensor de monezi |
| 3 | AT-1124-TWT-5V-2-R | difuzor |
| 4 | Arduino Mega 2560 | bazat pe ATmega2560 |
| 5 | Strip de Smart LEDs | RGB |
| 6 | FS-L-0055 | senzor de îndoire |
| 7 | EETI eGalaxTouch | sticlă cu touch |
| 8 | LCD | - |
| 9 | Releu | utilizat pentru Coin Hopper |
| 10 | Modul (MAX232) convertor de nivel TTL RS232 | pentru sticla cu touch |
| 11 | Convertor DC-DC | 12V → 5V/3A |
| 12 | Sursă de alimentare | 12V |

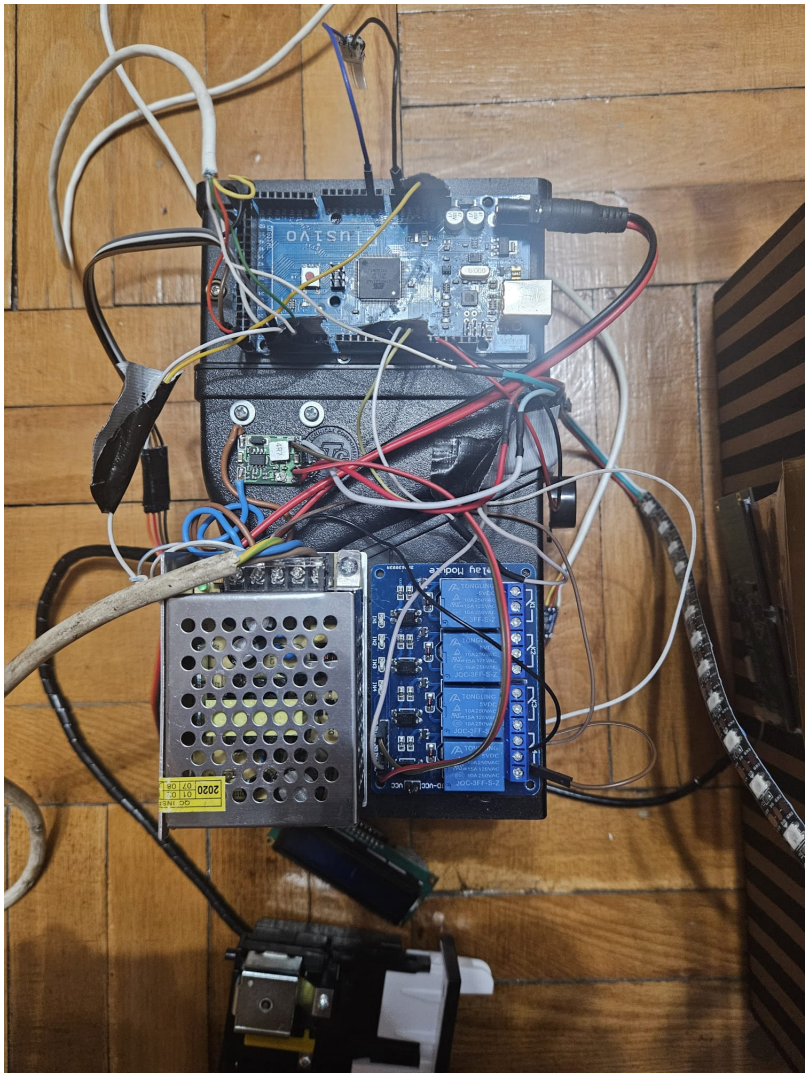
Conexiuni pini + detalii hardware:

- Acceptorul: conectat la 12V (sursa de alimentare), ground si pinii RX1/TX1 (ATmega2560: PD2/PD3, care au capabilitatea de a fi folosiți pentru RXD1/TXD1) (UART);
  - Comunică prin protocolul ccTalk, care folosește pinii RX și TX uniți pentru a realiza comunicarea pe un singur fir. Așadar, acceptorul are un pin comun care este conectat atât la RX-ul plăcuței, cât și la TX printr-o diodă și o rezistență de pull-up pentru a realiza configurația de multidrop

cerută de protocolul ccTalk. Bus-ul multidrop suportă mai multe device-uri, dar în acest proiect voi folosi doar acceptorul de fise (pentru că nu am hooper cu ccTalk)
- Dispensorul: conectat la 12V, prin releu; ground; conectat la pinul D9 al plăcuței (ATmega2560: PH6), prin care transmite câte un impuls pentru fiecare monedă pe care o scoate (GPIO).
  - Este conectat prin releu pentru a putea fi comandat de ieșirea din procesorul ATmega2560, care are capabilitate mică de curent (motorul dispensorului necesită un vârf de curent de 2A).
  - Este un model paralel foarte comun în industria de vending alături de ccTalk și MDB; functionarea lui se bazează pe antrenarea monedelor către ieșire cu ajutorul unui motor, numărarea acestora cu ajutorul unui senzor optic plasat pe calea de ieșire a monedelor și oprirea motorului în momentul în care suma de plată e atinsă; cât timp este alimentat, el scoate automat monede încontinuu.
- Releu: conectat la convertorul DC-DC (5V); ground; primește semnalul de on/off de la microcontroller prin pinul D8 al plăcuței (Atmega2560: PH5) pentru motorul dispensorului;
  - modelul folosit este quadruplu (dar este nevoie doar de o singură cale)
- Difuzor: conectat la ground și pinul D12 al plăcuței (ATmega2560: PB6, care are capabilitatea de a fi folosit pentru PWM), prin care este controlat (PWM);
  - difuzorul poate fi controlat direct de pinul de GPIO deoarece are o impedanță suficent de mare
- LCD: conectat la 5V de la plăcuță, ground și pinii SDA/SCL (ATmega2560: PD1/PD0, care au capabilitatea de a fi folositi pentru SDA/SCL) (I2C);
  - consumul de curent este suficient de mic încât să fie alimentat din LDO-ul intern al plăcii
- Smart LEDs (neopixeli): alimentați cu 5V de la convertorul DC-DC; ground; primește date (MOSI) de la pinul D51 al plăcuței (ATmega2560: PB2, care are capabilitatea de a fi folosit și pentru MOSI); restul pinilor nu există și nu sunt în uz (SPI);
  - Consumul de curent este foarte ridicat. La putere maximă poate consuma 60mA de fiecare LED RGB; folosim 40 de LED-uri, așadar se poate ajunge la 2.4A, așa că a fost necesară folosirea unui convertor DC-DC cu randament ridicat pentru a alimenta strip-ul de LED-uri.
- Sticlă cu touch controller: conectată la 5V la LDO-ul intern al plăcuței, ground și RX2/TX2 al plăcuței (ATmega2560: PH0/PH1, care au capabilitatea de a fi folosiți pentru RXD2/TXD2) (UART)
  - Deoarece senzorul tactil folosește nivele true RS232 (+/-12V), pentru Rx și TX, este necesar un modulul convertor de nivel pentru semnalele UART care sunt la nivele TTL. Acesta folosește un circuit comun în acest scop, MAX232. Intrările care nu sunt folosite au fost conectate la masă.
  - Folosește un protocol proprietar.
- Senzor de îndoire: conectat la ground și la pinul A1 al plăcuței (ATmega2560: PF1, care are capabilitatea de a fi folosit pentru ADC1) (ADC), și împreună cu un rezistor de pull-up de 50kOhmi formează un divizor de tensiune cu raport variabil. Divizorul de tensiune a fost conectat la 3.3V întrucât referința internă a ADC-ului este de maxim 3.3V și acesta nu poate citi valori mai mari.

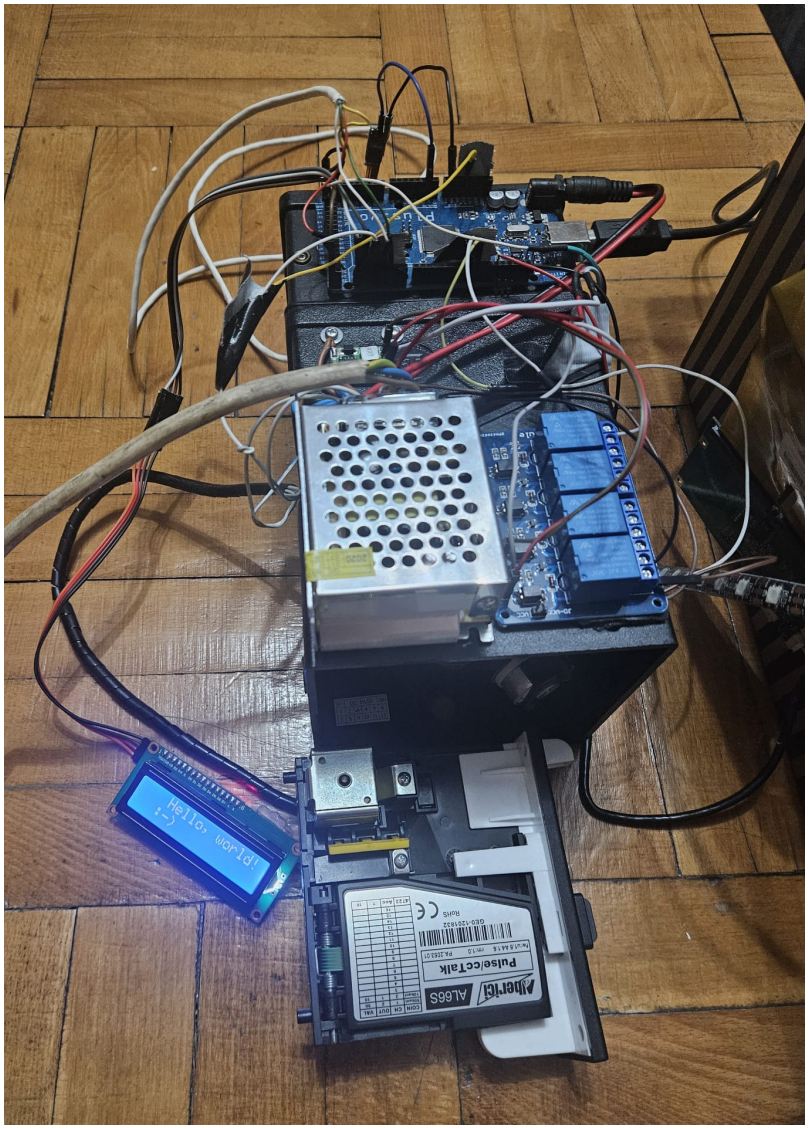
Imagini cu hardware-ul:





Dovadă funcționare:

Se poate vedea că LCD-ul este pornit și afișează textul dorit, ceea ce arată că microcontrollerul comunică corect cu acesta.




# Software Design

**Mediul de dezvoltare**: Proiectul a fost dezvoltat în mediul Visual Studio Code, folosind extensia PlatformIO și toolchain-ul avr-gcc.

**Librării și surse third party:**

| Header | Utilizare |
|---|---|
| <avr/io.h> | accesul la regiștrii microcontrollerului ATmega2560 |
| <util/delay.h> | funcțiile delay_ms, delay_us |
| <avr/interrupt.h> | sei(), cli(), macro-ul ISR() |
| <stdlib.h> | rand(), srand(), strtol(), etc (funcții ajutătoare legate de logica ruletei) |
| <string.h> | strlen(), strcpy(), etc (pentru procesarea comenzilor) |
| <ctype.h> | isdigit(), isspace(), tolower(), etc (pentru procesarea comenzilor) |
| <stdio.h> | snprintf() (formatarea mesajelor pe LCD) |
| <stdint.h> | tipurile de date |

| | |
|---|---|
| ”cctcom.h” | pentru comunicarea cu acceptorul folosind protocolul cctalk. sursă: https://cctalktutorial.wordpress.com/ |
| ”ws2812.h” | adaptat după librăria Adafruit Neopixel |

### Structura proiectului. Algoritmi și structuri folosite. Descrierea codului

- main.cpp: Conține majoritatea logicii proiectului, și bucla principală:
  - logica pentru jocul principal (bila care se rotește): compute_rotations, spin_leds_and_get_winner, etc
  - face trecerea între fazele jocului (welcome → choose bets + insert coins → spinning → dispense coins / show failure message)
  - conține funcția pentru interpretarea valorilor citite de la senzorul de îndoire: flex_is_outside_range. Interpretarea este minimală, deoarece nu este nevoie de date foarte concrete în logica legată de senzorul de îndoire (este folosit doar ca modalitate de acționare on / off; cât de îndoit este mai exact e doar ceva orientativ).
  - poll-uri pentru liniile seriale
  - logică de back-up pentru modalitatea de a efectua pariurile: citirea de la tastatură, pentru cazul în care se întâmplă ceva cu touch screen-ul :(
    - funcții precum normalize_token pentru parsarea datelor de la tastatură
- lcd_i2c.c / lcd_i2c.h: Fișierul conține funcțiile pentru i2c. Tot aici se găsesc funcțiile pentru comunicarea prin i2c cu LCD-ul 16×2. Funcțiile accesibile din acest fișier sunt lcd_init, lcd_backlight_on, lcd_backlight_off, lcd_clear, lcd_set_cursor și lcd_print. Se folosește expander PCF8574.
- uart.c / uart.h: Driver UART generic pentru USART0 (tastatură) / USART1 (acceptor) / USART2 (touch screen). Pune la dispoziție funcțiile uart_init, uart_available, uart_read, uart_write, folosite de altfel și în procesul de debugging. Toate primesc drept parametru baudrate-ul.
- timebase.c / timebase.h: Pentru funcția millis(). Conține funcțiile apelabile timebase_init si millis.
- acceptor.c / acceptor.h: Funcții pentru comunicarea cu acceptorul folosind protocolul ccTalk, adaptate din exemplele oferite pe https://cctalktutorial.wordpress.com/ , precum au fost și cctcom.cpp și cctcom.h. Conține acceptor_init, acceptor_sync_event_counter, acceptor_poll, cct_send_and_wait.
- adc.c / adc.h: Conține funcțiile de bază pentru adc: adc_init și adc_read.
- hopper.c / hopper.h: Conține funcții de bază pentru GPIO, și alte funcții legate de logica hopper-ului. hopper_on, hopper_off (acestea două teoretic interacționează cu releul), wait_for_level (asteapta sa primească de la hopper un semnal de monedă dispensată), dispense_coins. Logica dispensării monezilor e în hopper_run_payout + show_out_of_coins (se folosește de LCD).
- melody.c / melody.h: Vectorii pentru The Entertainer (melody + note_durations), set_note, set_silence, melody_start, melody_stop, melody_tick, wait_with_melody (pentru sincronizare), etc.
- pitches.h: Fișier pentru partea de muzică. Conține frecvențele sunetelor folosite, într-un range de 3 octave.
- pixels.c / pixels.h: Are funcțiile care se ocupă de întreg strip-ul de leduri. Funcțiile de aici se folosesc de cele din ws2812.h. Conține clear, init și set (pentru pixeli).
- touch.c / touch.h: Are funcții pentru pentru interacțiunea cu touch screen-ul: touch_map_symbol (face legatura dintre logica touch-ului și logica pariurilor), touch_in_rect (pentru detectarea zonei în care a fost atins ecranul), touch_poll_any și touch_poll_symbol (pentru detectarea atingerilor). Conține și partea de logică legată de detectarea tipului de pariu.
- ui.c: Conține funcțiile ce țin de interacțiunea cu omul prin LCD: lcd_print_line, show_new_bet_message, show_welcome_sequence.
- bets.c / bets.h: logica pentru pariuri, implementată aproximativ independent de modalitatea hardware prin care se efectuează aceste pariuri:
  - enum-ul SymbolType (pentru simbolurile pariurilor) și structura BetSymbol (pentru reprezentarea

abstractă a unui pariu)
- reset_bets: pentru resetarea pariurilor între jocuri
- vector pentru memorearea numerelor pentru care s-a pariat
- update_number, apply_symbol, is_black_number, count_selected_numbers, symbol_equal, find_symbol_index, etc: funcții ajutătoare pentru parsarea simbolurilor, detectarea numarului de numere pe care s-a pariat și care mai exact sunt acestea (logica principală din partea de pariuri)
- include funcții ajutătoare prentru integrarea LCD-ului, precum format_money, build_selection_line (linia cu ”SEL:” de pe ecran), bets_update_lcd_selection
- logică de back-up pentru modalitatea de a efectua pariurile de la tastatură: funcții precum parse_symbol pentru interpretarea datelor de la tastatură

Detalii suplimentare: Funcționalitatea proiectului a fost validată prin testări succesive pe parcursul implementării, și realizarea separată a codului pentru fiecare modul hardware în parte, urmată de testarea interacțiunii cu acesta înainte de integrarea codului în proiectul mare. Singurul senzor este cel de îndoire, și valorile între care operează acesta au fost detectate la începutul implementării printr-o bucată de cod separată (program individual) și apoi integrate în restul proiectului. Link video: https://drive.google.com/file/d/1YxeMvDDyLzt8izYvN4p_wdTzLu7mX2mg/view?usp=drive_link

# Rezultate Obţinute

Rezultatele obținute: o ruletă funcțională, poate puțin instabilă structural (din cauza touch screen-ului care trebuie proptit vertical), ce poate fi folosită pentru a juca codul de acasă. Nu e nevoie să fie conectată la laptop pentru a juca jocul, are sursă de alimentare externă așa că poate fi pur și simplu băgată în priză și folosită.

Rezultat final:


Acceptor + dispensor:

Senzor de îndoire:


## Concluzii

În concluzie, acest proiect a fost mult mai complicat decât mă așteptam să fie, în principal din punctul de vedere al codului (dar nici hardware-ul nu se lasă mult mai prejos). Altă concluzie ar fi că este o diferență foarte mare între ceva funcțional, ceva utilizabil și un joc care îți face plăcere să îl folosești. Aș spune că în momentul de față proiectul este funcțional, dar sunt multe alte feature-uri care mi-ar fi făcut plăcere să am timp să le adaug, precum mai multe sunete (melodie pentru lose, sunet de ping atunci cand se rotește luminița), o stare de idle în care să treacă aparatul atunci când nu e folosit mai mult timp la rând (cu o melodie de idle și eventual un dans pe care să îl facă ledurile), o modalitate de a forța norocul numerelor într-o direcție sau alta (cu mare părere de rău, nu am avut timp să fac asta,

așa că aceasta este o ruletă onestă). Cu toate acestea, mi-a făcut plăcere să am în față un produs finit, funcțional.

## Download

Codul: munteanu_andreea.zip

## Bibliografie/Resurse

Resurse Hardware: datasheet-urile componentelor folosite

- ATmega2560
- acceptor
- dispensor
- senzor de îndoire

Resurse Software:

- CCTalkTutorial
- librăria AdafruitNeopixel pentru informații legate de modul de comunicare cu ledurile

Export to PDF