POLITECHNIKA ŚLĄSKA

WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI

Projekt z Systemów Mikroprocesorowych

Jednoręki Bandyta

Autorzy: Jakub Kula & Gabriel Wyrzychowski

2022-2023, Semestr 5, grupa 6, sekcja 2

Kierujący pracą: dr inż. Krzysztof Jaskot

Gliwice, styczeń 2023

1. Spis treści

1. Wstęp

1.1. Cel i zakres projektu

Jednoręki bandyta jest projektem którego głównym założeniem jest odtworzenie konceptu klasycznej maszyny hazardowej znanej właśnie jako „jednoręki bandyta" przy użyciu mikroprocesorów jednocześnie będącym pierwszym kontaktem jego twórców z tworzeniem prostych układów elektrycznych oraz programowaniem wcześniej wspomnianych mikroprocesorów. Zakładane jest wykorzystanie jednego mikroprocesora, jednego wyświetlacza oraz przycisków w ilości potrzebnej to w pełni prawidłowego funkcjonowania (końcowa ilość została ustalona na trzy sztuki). Wyświetlacz ze swej natury służy do wyświetlania informacji koniecznych dla gracza, takich jak posiadana ilość waluty, informacja o wygranej bądź przegranej, instrukcja, informacje odnośnie zmiany wielkości zakładu, minigra pozwalająca na zarabianie waluty. Przyciski będą służyć do „poruszania" się po menu jak i do ogólnego zatwierdzania ustawień czy też wyników. Projekt nie ma na celu stworzenia prawdziwej maszyny która mogłaby faktycznie zostać wykorzystana w hazardzie, a mikroprocesorowego zabawkowego odpowiednika którego głównym zadaniem będzie bycie pewnego rodzaju prototypem dla dalszych konceptów, jak i po prostu grą w którą można zagrać dla czystej rozrywki.

1.2. Wymagania sprzętowe

Ogólne wymagania postawione przed doborem sprzętu są bardzo proste w założeniach: powinien dać radę spełnić nasze oczekiwania. Na początku tworzenia projektu nie było możliwości precyzyjnego ocenienia wymogów, gdyż występował wyraźny brak doświadczenia u autorów, więc wymagania rozwijały się wraz z rozwojem projektu. Pierwotnym założeniem było wykorzystanie płytki Arduino Uno z zawartym w niej Atmega328, gdyż jest to sprzęt w przystępnej cenie oraz wprost stworzony dla nowicjuszy. Następnie było rozpatrywanie użycie Raspberry Pi, jednakże z przyczyn opisanych w dalszych częściach raportu pomysł ten został ostatecznie odrzucony. W przypadku wyświetlacza zdecydowaliśmy się na wyświetlacz RGB Waveshare 13892 gdyż posiada on odpowiednie wymiary oraz kolory. Chwilowo jako alternatywa były rozpatrywane wyświetlacze LCD, jednak końcowo nie przeszły dalej niż bycie luźnym pomysłem. W przypadku przycisków są to Tact Switch'e- bardzo standardowa i podstawowa opcja. Inne opcje nie były rozpatrywane gdyż nie było zapotrzebowania wynikającego z wymagań sprzętowych.

2. Harmonogram

2.1. Harmonogram zatwierdzony

Zajęcia nr 1-Próby zapoznania się z hardwarem oraz stworzenie ekranu startowego.

Zajęcia nr 2-Stworzenie modułu losowania. Zaimplementowanie przycisków, modułu menu, sterowania po nim. W tym możliwości zmniejszania i zwiększania wartości BID oraz możliwości ponownej gry.

Zajęcia nr 3-Dodanie animacji losowania oraz jackpota, napisanie instrukcji.

Zajęcia nr 4-Podłączenie głośnika oraz zaprogramowanie krótkiego dźwięku, który będzie grany podczas wygranej

Zajęcia nr 5-Dopracowywanie projektu, między innymi ulepszenie wyświetlenia wartości punktów, BIDu oraz wygranej. Optymalizacja kodu w celu zmniejszenia zużytej pamięci. Testy oraz naprawa błędów

2.2. Harmonogram wykonany

Zajęcia nr 1-Próby zapoznania się z hardwarem oraz stworzenie ekranu startowego.

Zajęcia nr 2-Stworzenie modułu losowania. Zaimplementowanie przycisków, modułu menu, sterowania po nim. W tym możliwości zmniejszania i zwiększania wartości BID oraz możliwości ponownej gry.

Zajęcia nr 3-Dodanie animacji losowania oraz jackpota, napisanie instrukcji.

Zajęcia nr 4-Stworzenie możliwości zdobycia waluty innej niż główna.

Zajęcia nr 5-Dopracowywanie detali projektu, między innymi ulepszenie wyświetlenia wartości punktów, BIDu oraz wygranej. Optymalizacja kodu w celu zmniejszenia zużytej pamięci. Testy oraz naprawa błędów

Zdecydowano się wymienić moduł dźwięku w zamian na możliwość zdobycia waluty w inny sposób, ponieważ rozwiązuje to problem, w którym gracz przegrał cała i nie jest w stanie dalej grać, oraz w dłuższym posiedzeniu dźwięk wygrywania mógłby być bardzo denerwujący dla gracza.

3. Kosztorys

Koszty poniesione w ramach stworzenia projektu wskazane są w poniższej tabeli.

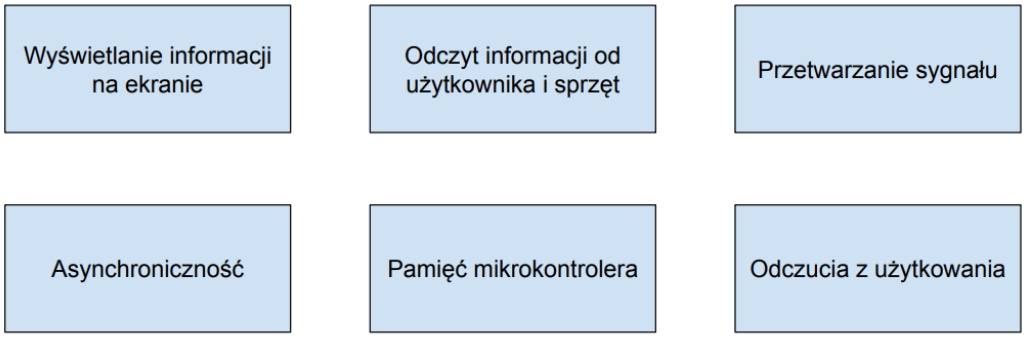
LP	Typ	Ilość	Cena [zł]	Wartość [zł]
1	Arduino Uno	1	34,36	34,36
2	Przycisk	3	0,50	1,50
3	Przewody	n/e	n/e	3,00
4	Tft display waveshare 13892	1	39,00	39,90
5	Obudowa	1	16,10	16,10
6	Płytka stykowa	1	6,00	6,00
SUMA				100,86

Czas przeznaczony na naukę oraz właściwą pracę nad projektem jest szacowany na około 45 godzin.

4. Urządzenie wraz z aplikacją

4.1. Określenie i analiza problemu

Dokładne opisanie problemu. Wykonać schematy ideowe i blokowe, dołączyć ilustrujące problem rysunki itp. Wykonać założenia potrzebne do rozwiązania postawionego problemu.



Rysunek 1. Schemat blokowy problemów.

Pierwszym problemem zaznaczonym w 4.1 jest wyświetlanie obrazu, musi być ono czytelne, przyjemne dla oka oraz ciekawe. Tak aby użytkownik czuł przyjemność z gry i nie był zniechęcony do użytkowania naszego projektu

Odczyt informacji od użytkownika powinien być taki jak w klasycznych maszynach arkadowych, prosty tak aby każdy mógł cieszyć się grą już od samego początku, a ekran powinien być wystarczająco duży i czytelny, oraz którego cena będzie odpowiednia do naszego budżetu

Przetwarzanie sygnału powinno odbywać się przez nasz system mikroprocesorowy, powinno zostać użyte wiele napisanych funkcji ponieważ oszacowano ze wiele czynności takich jak wyświetlanie aktualnego stanu ilości waluty czy wysokość BIDu będzie zaimplementowany wielokrotnie.

Po dokonaniu korekcji pomysłu, jednym z naszych wyzwań będzie próba stworzenia sztucznej synchroniczności – tak aby system był w stanie czekać odpowiednią ilość czasu oraz czasu, ale przerwał to w momencie dostania sygnału od gracza. A użyty przez nas mikrokontroler jest asynchroniczny

Ilość pamięci kontrolera jest bardzo ograniczona, przez co kod będzie musiał być zoptymalizowany. Ilość pamięci jaką dysponuje mikrokontroler Atmega328 to tylko 2 KB pamięci SRAM, 32KB Flash oraz 1KB EEPROM .

Najważniejszym problemem jest zmaksymalizowanie przyjemności grania. Niestety, dla naszego projektu nie możemy zastosować klasycznego ramienia jak z oryginalnych jednorękich bandytów, gdyż stanowiło by to problem w płaszczyźnie tworzenia oprogramowania, gdyż takie ramię nie jest szczególnie standardowym elementem współczesnej elektroniki, jak i fizycznie wiązałoby się to ze zwiększeniem gabarytów całego projektu. Oba te czynniki dodatkowo nakładają się z ograniczonym budżetem, co całkowicie odrzuca możliwość instalacji ramienia. Z tego powodu musieliśmy znaleźć alternatywę pod postacią przycisku.

4.2. Rozwiązanie problemu

4.2.1. Wyświetlanie informacji na ekranie

Wyświetlacz Waveshare 13892 jest obsługiwany przez bibliotekę Adafruit_GFX.h która dostarcza nam podstawowe funkcje do wyświetlania obrazu, takie jak:

- void drawPixel(uint16_t x, uint16_t y, uint16_t color) – funkcja która zmienia na wskazanych koordynatach x, y kolor piksela na color
- void drawLine(uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1, uint16_t color) – pozwala nam narysować linie która zaczyna się w x_o, y_o a kończy się w x_t, y_t o kolorze color
- void drawRect(uint16_t x0, uint16_t y0, uint16_t w, uint16_t h, uint16_t color) – dzięki tej funkcji narysujemy prostokąt o początku w x0, y0 wysokości h i długości t w kolorze color.
- void drawBitmap(int16_t x, int16_t y, uint8_t *bitmap, int16_t w, int16_t h, uint16_t color) będzie to przez nas najczęściej używana funkcja ponieważ daje nam ona możliwość wyświetlania bitmapy o nazwie bitmap w koordynatach x, y oraz wielkości h, t w kolorze color, co będzie naszym największym ograniczeniem ponieważ każdą figurę będziemy musieli rozbić na pomniejsze które będą zawierać tylko jeden kolor, lub rysować pojedyncze piksele w odpowiednich miejscach

To są główne funkcje które będą nam służyć do wyświetlania obrazku w naszym projekcie, choć biblioteka daje nam znacznie więcej możliwości takich jak np. rysowanie okręgu czy trójkąta.

Niestety pisanie tekstu nie będzie aż tak proste jak wyświetlanie figur, ponieważ każdy napisany tekst będziemy musieli poprzedzić ustawieniem kursora w odpowiednim miejscu przy użyciu **void setCursor(int16_t x0, int16_t y0)** który pozycjonuje początek napisane tekstu na koordynaty x0, y0. Dopiero po tym będziemy w stanie edytować to jak wygląda nasz tekst przy użyciu

- void setTextColor(uint16_t color) – dzięki ustawieniu na początku koloru naszego tekstu na czerwono mogliśmy nadać naszemu projektowi bardziej kasynowego wyglądu
- void setTextSize(uint8_t size) – który ustawia nam wielkość tekstu, niestety w naszym przypadku najczęściej ograniczało się to wielkości 1.
- void setTextWrap(boolean w) – pozwala nam automatycznie przerzucić tekst do kolejnej linijki, niestety ta funkcja nie działała tak jak byśmy chcieli i zostawiała bardzo duże czarne odstępy na końcach linijek

Jednak same funkcje nie wystarczyły aby nasz projekt wyglądał jak zamierzano. Zaprojektowaliśmy w tym celu 5 pixel artów w rozdzielczości 16x16 które byty stylizowane na te które możemy znaleźć w klasycznym odpowiedniku.

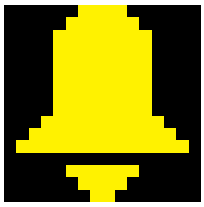


Figura nr.1– dzwon

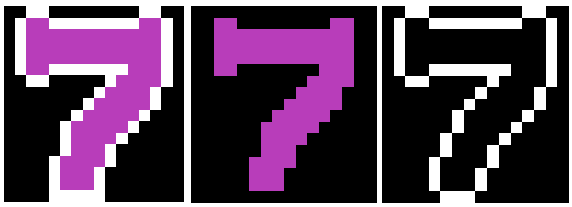


Figura nr.2 – liczba siedem i jej obramowanie

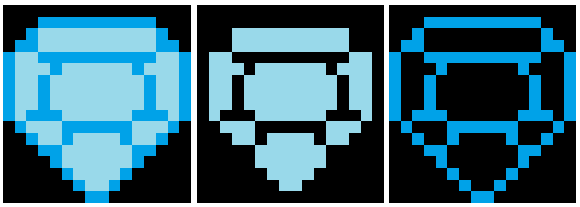


Figura nr.3 – diament i jego obramowanie

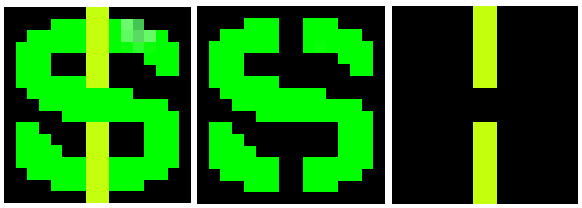


Figura nr. 3 – Dolar i jego szczegóły

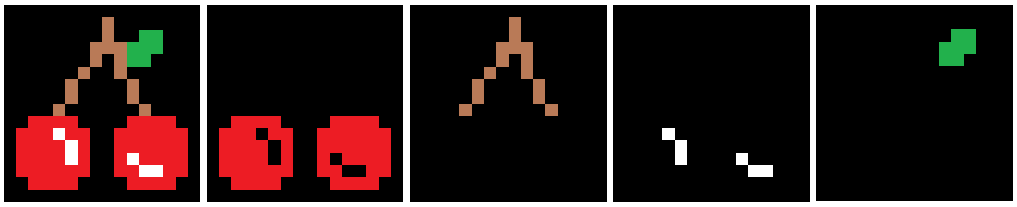


Figura nr.4 – Wiśnia oraz jej elementy składowe

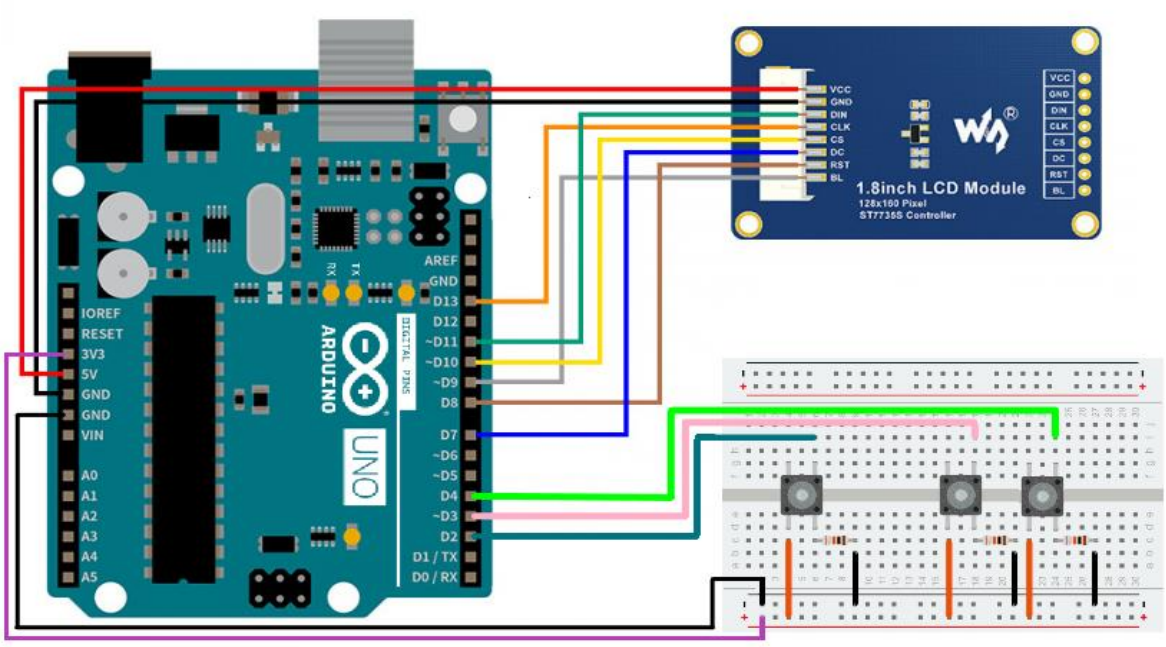
Jak można zauważyć wszystkie nasze bitmapy musiały zostać podzielone na osobne pliki w zależności od kolorów, nie licząc czarnego tła. Zostało to zastosowane przez ograniczenia sprzętowe. Pierwszym ograniczeniem była strona[4] dzięki której mogliśmy zmienić plik bitmapy na bit array. Jednak tło musiało być w jednolitym kolorze białym lub czarnym lub przeźroczystym. Natomiast drugim ograniczeniem była funkcja wyświetlająca bitmapę, ponieważ jest ona w stanie wyświetlić obraz jedynie w jednym kolorze, wiec takie podejście było od nas wymagane.

4.2.2. Odczyt informacji od użytkownika i sprzęt

Spośród całej gamy różnych ekranów został wybrany Waveshare 13892, gdyż spełniał on nasze wszystkie kryteria,

- Wystarczająco duży 1,8” 128 na 160 pikseli co dało nam wystarczająco dużo miejsca na ekranie do wyświetlanie wszystkiego co było zaplanowane.
- Kolorowy wyświetlacz RGB 65K byliśmy świadomi że będzie on trudniejszy do programowania niż jego niebieski odpowiednik jednak wiedzieliśmy że efekt będzie znacząco lepszy.
- Łatwy w użytkowaniu – zdawaliśmy sobie sprawę ze nawet przy wystąpieniu jakichkolwiek problemów będziemy w stanie znaleźć duża ilość materiałów pomocniczych w Internecie.

Wybrany przez nas kontroler to Atmega328 zawarty w Arduino uno ze względu na prostość programowania oraz posiadane przez nas minimalne obycie które uzyskaliśmy na laboratoriach.



Rys nr 1. Schemat projektu[5][6]

Do portów 2, 3, 4 zostały podpięte wyjścia przycisków, a do ich wejścia zostało podpięte napięcie 3.3V oraz GND. Natomiast nasz ekran jest zasilany 5V, tak aby działał poprawnie, choć powinien on tez działać przy napięciu 3.3V. Reszta portów naszego wyświetlacza:

- CS – chip select, który jest głównie używany przy połączeniu większej ilości urządzeń
- RST – Reset, jest on odpowiedzialny za reset naszego wyświetlacza który jest potrzebny podczas inicjalizacji
- DC – Data/Command odpowiada za wysyłanie danych i poleceń do wyświetlacza

4.2.3. Przetwarzanie sygnału

Głównym założeniem przetwarzania sygnału było zainicjowanie trzech głównych zmiennych lokalnych:

- BID – Wartość zakładu która jest odbierana podczas w momencie wejścia do losowania.
- Punkty – wartość waluty jaka posiada gracz w danym momencie, od której będzie odbierana wartość BID w momencie zagrania w jednorękiego bandytę, a dodawana po wygranej.
- MENU – jest to zmienna pomocnicza używana do sterowania menu po wciśnięciu którekolwiek z przycisków

Po włączeniu się mikrokontrolera będą pokazywać się wstępne informacje o projekcie takie jak nazwiska twórców, nazwa projektu oraz napis wydziału, grupy i semestru
Następnie zostanie nam wyświetlony ekran powitalny, oraz gracz zostanie poinformowany o początkowej ilości waluty. Między tymi ekranami jest 1.5s przerwy

Zdjęcia nr 1 i 2 pokazujące wyświetlaną nazwę projektu oraz ekran powitalny

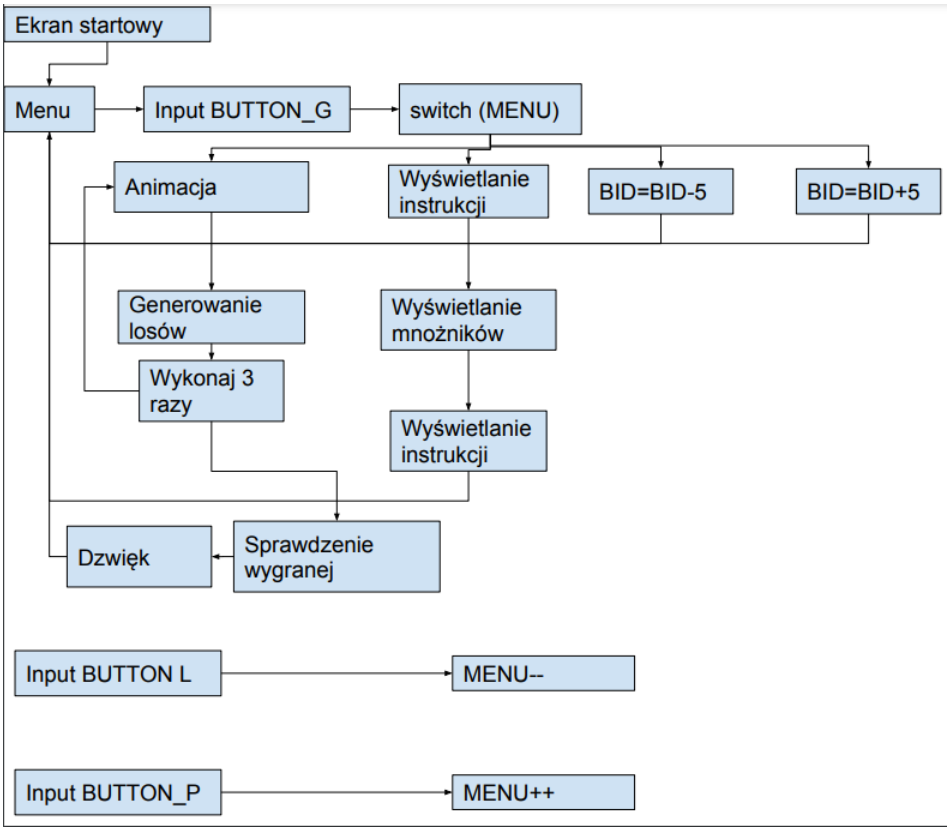
Po sekwencji startowej, gracz zostanie przeniesiony do menu głównej gdzie aplikacja będzie czekać na kliknięcie przycisku przez gracza. W zależności od wciśniętego przycisku mogą stać się 3 rzeczy:

- Wciśniecie przycisku lewego spowoduje zmniejszenie się wartości zmiennej globalnej MENU o jeden, następnie nasze menu zostanie odświeżone tak aby pokazywać możliwość która jest aktualnie wybrana. Jeżeli wartość MENU miała by osiągnąć wartość zera, zostaje automatycznie ustawione na wartość maksymalną.
- Wciśniecie przycisku Prawego spowoduje rzecz analogiczną rzecz do przycisk lewy z tą różnicą że MENU zostanie zwiększone
- Wciśniecie przycisku głównego powoduje że wchodzimy do instrukcji switch która jest zależna od parametru MENU.

Gdy wartość MENU jest równa jeden a napis na dole ekranu wyświetla napis „Jednoręki bandyta" i zostanie wciśnięty przycisk główny, następuje odebranie od ilości punktów wartości BID na poczet gry, jeżeli wartość punktów jest większą od niego, jeżeli nie, zostanie to przekazane użytkownikowi na górze ekranu. Następnie na podstawie czasu włączenia systemu mikroprocesorowego zostanie wygenerowany seed, po czym zostaną wylosowane trzy liczby w przedziale od 1 do 5. W odpowiednim miejscu na ekranie pojawi się animacja, a po niej zostaną wyświetlone wylosowane figury. Na samym końcu zostanie sprawdzenie wygranej, wyświetlenie ilość wygranej waluty oraz w początkowej wersji miała być zagrana muzyka w zależności od wysokości wygranej.

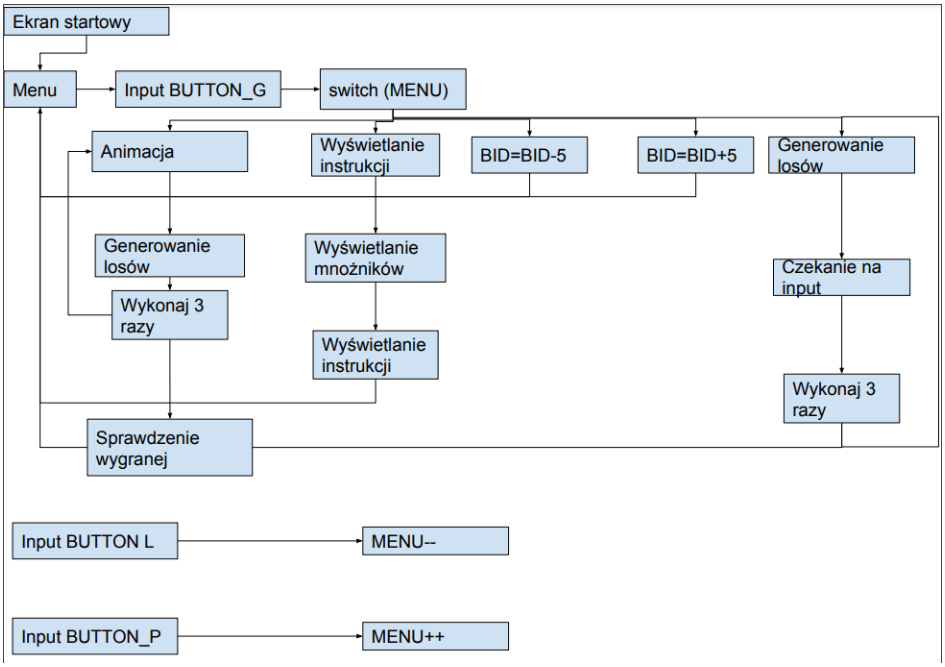
Natomiast jeżeli wartość MENU będzie równa 2, instrukcja switch przekieruje nas na możliwość zwiększenia wartości BIDu o pięć. Analogicznie będzie dla wartości równej 3 tyle ze zmniejszy ona BID.

Wyświetlanie instrukcji jest podzielone na 2 etapy, gdzie pierwszy i ostatni to wyświetlanie bloku tekstu opisującego zasady gry a środkowy to wypisanie mnożników jakie dają poszczególne pary figur. Wyświetlanie każdego z etapów odbywa się do wciśnięcia przycisku głównego przez grającego



Rys nr 2. Schemat blokowy na początku projektu

Jednak po konsultacjach nasz schemat blokowy musiał zostać zmieniony, została dodana kolejna opcja, w instrukcji switch oraz jest to pozbycie się modułu dźwięku.



Rys nr 3. Schemat blokowy na końcu projektu

Nowy moduł bazuje na bardzo podobnej zasadzie co nasz główne. Najpierw następuje usuniecie poprzedniej figury, następnie losowanie nowej i następuje pętla czekania z możliwością przerwania jej, po której jest generowany nowy seed na podstawie czasu. Cały proces jest powtarzany trzy razy, aby kolejno sprawdzić czy dany zestaw figur jest wygrywający. Po wyświetleniu wyniku następuje opóźnienie trwające 1.7s i zostajemy przeniesieni do menu głównego.

Zdjęcie nr 3 przedstawiające efekt (nieudanego) losowania

4.2.4. Asynchroniczność

Wraz z nowym pomysłem zastąpienia modułu dźwięku modułem możliwości zdobycia waluty pojawił się nowy problem i była nim brak asynchroniczności w Atmega328. A był on nam potrzebny aby po każdym losowaniu można było dać użytkownikowi czas na zareagowania i zatrzymanie go na wymarzonej figurze. Zainspirowani naszym ostatnim projektem na laboratoria z przedmiotu Systemy operacyjne, zrobiliśmy pętle obciążenia, która poza zwiększaniem się naszej liczby, sprawdzała też czy przycisk główny jest wciśnięty.

```
while (true)
{
  tft.fillRect(20, 73, 18, 18, ST77XX_BLACK);
  Los_1 = random(1, MAX);
  figury(-35, Los_1);
  j = 0;
  time = millis() * 12.234561;
  randomSeed(time);
  while (j < TIME)
  {
    j++;
    if (digitalRead(BUTTON_G) == !LOW)
    {
      flaga = 1;
      delay(200);
      break;
    }
  }
  if (flaga)
    break;
}
```

Wspomniana wyżej pętla obciążenia

Jak możemy zauważyć nowe seed jest generowany zaraz po wyświetleniu starej figury jednak nie ma to znaczenia, ponieważ czas reakcji człowieka waha się od 0.3s aż do nawet 1.7s. Rzecz na która warto zwrócić uwagę jest stała ST77XX_BLACK, jest to jedna ze stałych która została przygotowana przez twórców która odpowiada kolorowi czarnemu w systemie heksadecymalnym.

4.2.5. Pamięć mikrokontrolera

Pamięć kontrolera Atmega328 to tylko 2 KB pamięci SRAM, 32KB Flash oraz 1KB EEPROM, wiec musieliśmy zważać na to na każdym etapie pisania kodu. Jednym z pierwszych modyfikacji które poczyniliśmy było zastąpienie pisania każdej z naszej figur piksel po pikselu, na bit array, gdyż to rozważanie zajmuje ponad 10 krotnie mniej pamięci niż poprzednie. Oraz było ono znacząco szybsze i przyjemniejsze. Po własnoręcznym narysowanie każdego z pixel artów, zostały one podzielone na pomniejsze pliki, gdzie każdy z nich zawierał tylko jeden kolor. Następnie została użyta strona NUMER ZAŁĄCZNIKA która pozwoliła bardzo szybko skonwertować bitmapę na bitarray.

```
const unsigned char bell[] PROGMEM = {
0x03, 0xc0, 0x07, 0xe0, 0x0f, 0xf0, 0x0f, 0xf0, 0x0f, 0xf0, 0x0f, 0xf0, 0x0f, 0xf0, 0x0f, 0xf0,
0x0f, 0xf0, 0x1f, 0xf8, 0x3f, 0xfc, 0x7f, 0xfe, 0x00, 0x00, 0x07, 0xe0, 0x03, 0xc0, 0x01, 0x80};
```

Przykładowy zapis jednej z figur

Jednak to rozwiązanie też miało swoje minusy. Każda z figur była zapisywana w pamięci podręcznej, czego chcielibyśmy uniknąć. Aby rozwiązać ten problem użyliśmy dyrektywy „PROGMEM" – która jak się dowiedzieliśmy działa najlepiej na zmiennych globalnych. Dzięki niej nasze tablice były przechowywane w pamięci flash.

Udało nam się także zmniejszyć ilość linijek kodu odpowiedzialnych za wyświetlanie punktów, BIDu oraz wygranej, dzięki zauważeniu że każda z cyfr potrzebuje tylko czterech pikseli szerokości oraz po jednym na każdej ze stron aby oddzielić je od reszty. Dzięki temu zaoszczędziliśmy miejsce gdyż ustawienie kursora początku linii było zależne od wielkości liczby, Przykładowo zmienna BID była przesuwna w prawo o wielokrotność trzech pikseli a znaczek $ trzy w prawo

```
    if (bid <= 9)
    i = 0;
  else if (bid <= 99)
    i = 1;
  else if (bid <= 999)
    i = 2;
  else
    i = 3;
  tft.setCursor(10, 0);
  tft.setTextSize(1);
  tft.println("Bid: ");
  tft.setCursor(14 - 3 * i, 10);
  tft.println(bid);
  tft.setCursor(20 + 3 * i, 10);
  tft.println("$");
```

Wiedząc że niektóre moduły będą używane wielokrotnie w naszym programie udało nam się napisać kilka funkcji które powodują że kod staje się czytelniejszy i zajmuje mniej miejsca:

- display_punkty()
- display_bid()
- display_menu()
- wygrana(int j. int d, int t)
- animacja(int p)
- figura(int p, int f)

Gdzie pierwsze trzy funkcje służą nam do wyświetlania sekcji informacyjnej projektu, wygrana sprawdza czy trzy podane wartości są takie same i oblicza ile punktów dodać i jest ona używana w głównej części naszego projektu oraz w dodatkowym module zdobywania waluty. Animacja oraz figury służą nam do wyświetlania naszych bitmap w odpowiednim miejscu.

Dzięki zastosowaniu tych rzecz udało nam się zająć tylko 20610 bajtów przeznaczonych na pamięć programu co stanowi 63% całości oraz 1276 pamięci dynamicznej pozostawiając nam jeszcze 772 bajty na zmienne. Co daje nam jeszcze dość duże pole do ulepszania działania kodu oraz dodawania nowych modułów.

4.2.6. Odczucia użytkownika

Projekt został przedstawiony jednej niezależnej osobie, aby przetestowała jego możliwości. Uzyskane informacje zwrotne sugerowały potrzebę wyjaśnienia jak należy się poruszać po menu oraz który przycisk jest zatwierdzającym. Spowodowało to pewne modyfikacje w wyświetlanej instrukcji jak i w obrębie fizycznej oprawy projektu. Dodatkowo z uzyskanej opinii wynika że jednoręki bandyta spełnia swoją funkcję jako małe urządzenie do rozrywki.

4.2.7. Matematyka wygranej

Odczucia użytkownika były naszym priorytetem dlatego też w tym przypadku kasyno nie wygrywa zawsze. Sposób wygranej został zaprojektowany w taki sposób żeby gracz miał przewagę nad urządzeniem miedzy innymi poprzez użycie:

- Tylko pięciu figur co powoduje ze szansa na wygranie jesteśmy w stanie opisać wzorem:

$$P(W) = \frac{|W|}{|\Omega|} = \frac{5 + 5 * 4 * 3}{5^3} = 52\%$$

Gdzie:

|W| - to liczba zdarzeń sprzyjających, ilość możliwych kombinacji które zdobywają walutę, w naszym przypadku jest to 5 możliwości zdobycia trzech takich samych figur oraz 5 możliwości zdobycia dwóch takich samych figur pomnożone przez 3 możliwości ułożenia tych figur oraz 4 możliwości dodatkowej figury.

|Ω| - to liczba wszystkich możliwych zdarzeń

- Każdy jackpot niezależnie od figury daje nam jedenastokrotność BIDU niezależnie od figury, oraz to ze każda figura ma swój mnożnik można policzyć wartość oczekiwaną która będzie równa:

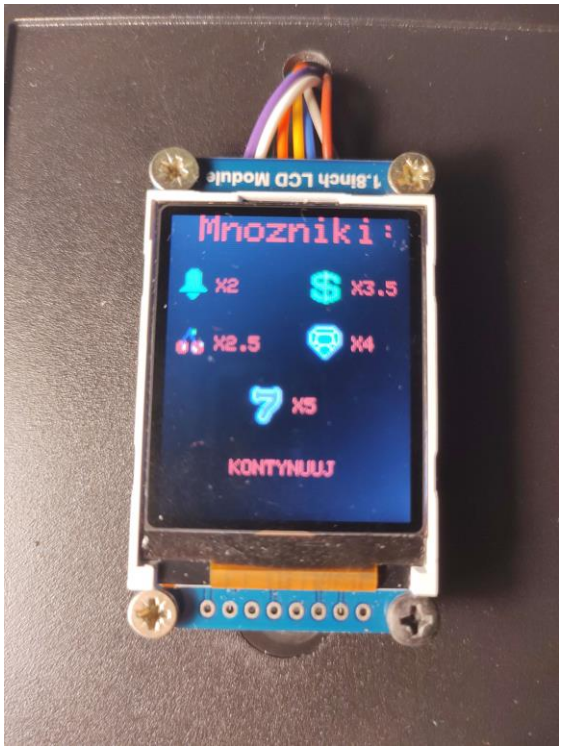
$$E(W) = \sum_{i=1}^{n} W_i p_i = 0 * 36 + \frac{11}{25} + \frac{6}{25} + \frac{3}{10} + \frac{21}{50} + \frac{12}{25} + \frac{3}{5} = 2$$

Gdzie:

W_i – Wartość wygranej wydarzenia

p_i – Prawdopodobieństwo wystąpienia zdarzenia W

Do obliczeń zostało przyjęte ze szansa na zdobycie jedenastokrotności ilości BIDU jest równa 4% a zdobycia podwójnie jednej figury to 12%. Mnożniki dla poszczególnych figur pokazane są na poniższym zdjęciu.



Zdjęcie nr 4 przedstawiające poszczególnym mnożników wygranej

Po przeanalizowaniu wartość oczekiwaną ale szanse na wygraną stwierdzono, że jest ona wyższa niż 50% oraz „1”, wiec kasyno w tym przypadku nie ma żadnej przewagi nad graczem jak w standardowym kasynie.

4.2.8. Animacje

Animacje odgrywają bardzo ważna role grze, gdyż nadają one jej dynamikę, dlatego zdecydowano się przyjąć pewne opóźnienie miedzy każdą akcją. W większości przypadków przyjęto to opóźnienie rzędu od 0.2s do 0.3s. Można je zaobserwować miedzy innymi przy próbie zwiększenia BIDu albo zmiany opcji w menu. Dzięki zastosowaniu tego, gracz jest w stanie zauważyć zmianę która dokonał poprzez wciśnięcie przycisku, czuje ze jego akcje mają przełożenie na wyświetlające się rzeczy na ekranie, oraz pozbyto się możliwości ze pojedyncze wciśniecie przycisku będzie interpretowane jak wiele. Sposób implementacji animacji podczas losowania figur był inspirowany starymi maszynami arkadowymi.

```
for (int i = 50; i <= 160; i = i + 8)
{
  time = millis() * 12.234561;
  randomSeed(time);
  Los_s = random(1, MAX);

  if (Los_s == Los_poprzedni)
  {
    time = millis() * 986;
    randomSeed(time);
    Los_s = random(1, MAX);
  }
  figury(p, Los_s);
  Los_poprzedni = Los_s;
  delay(i);
  tft.fillRect(55 + p, 73, 18, 18, ST77XX_BLACK);
}
```

Pętla animacji losowania figur

Cała animacja jest zawarta w jednej pętli „for” dla każdej z pozycji, gdzie na początku jest tworzony seed na podstawie czasu włączenia mikrokontrolera, następnie następuje losowanie figury i sprawdzenie czy nie jest taki jak poprzedni, gdy jest, następuje zmiana. Następnie figura zostaje wyświetlana i czekamy 50ms, lecz za każdą iteracją czas po wyświetlaniu zostaje zwiększony 8ms. Tak aby każde kolejne wyświetlanie trwało dłużej. Czyli tak jak było to zrobione w starych maszynach.

4.3. Problemy w trakcie tworzenia sprzętu i aplikacji

4.3.1. Hardware

Podczas tworzenia tego projektu napotkaliśmy wiele problemów, pierwszy z nich pojawił się już na samym początku wraz z próbą użytkowania Raspberry pico, gdzie mimo instalowania potrzebnych bibliotek, niektóre z nich nie chciały działać.

Niestety po wielu godzinach próby rozwiązania tego problemu, zmieniliśmy podejście i sprawdziliśmy czy nasz ekran działa na innym mikrokontrolerze którego sprawności byliśmy pewni. Po podłączeniu zasilania do wejście 3.3 V zauważyliśmy że wygląd naszego ekranu odbiegał od tych które mogliśmy znaleźć w Internecie. Jednak po przepięciu na 5 V problem ustał.

Więc przystąpiliśmy do prób sprawdzenia działania naszego Raspberry. Po bardzo prostych testach zauważyliśmy ze komunikacja z mikrokontrolerem przeszła pomyślnie. Wiec po konsultacjach podjęliśmy decyzje o przylutowaniu goldpinów do mikrokontrolera. Co rozwiązało nasz problem z migającym ekranem jednak nie pomogło nam z największym problemem. Po kolejnych godzinach pracy, zmieniliśmy nasz mikrokontroler na Arduino Uno. Jednak wiązało się to z zmniejszeniem pamięci która mogliśmy rozdysponować. Z tego powodu nasze programowanie musiało być bardzo przemyślane już od samego początku.

4.3.2. Software

Największym problemem jaki napotkaliśmy podczas pisania naszego kodu było pozycjonowanie wszystkich rzecz na ekranie. Tak aby wyglądało to estetycznie i miło dla oka. Zdecydowaliśmy się wyświetlać aktualny BID w lewym górnym roku a punkty w prawym. Następnie poniżej jest wyświetlany ostatni wynik gry w Jednorękiego bandytę, jeśli jest to niemożliwe, są wyświetlane trzy figury siedem. A poniżej aktualnie wybrana opcja menu.

5. Podsumowanie

5.1. Elementy zrealizowane

Z pierwotnych założeń zrealizowaliśmy zdecydowaną większość elementów. Jednoręki bandyta symuluje grę hazardową opartą o wylosowanie kombinacji trzech identycznych znaków jako warunek wygranej. Zaimplementowany został wyświetlacz, na którym wyświetlane jest kilka rodzajów obrazków. Jednoręki bandyta posiada trzy funkcjonalne przyciski obsługujące menu, które z kolei zawiera grę, instrukcję, mini-grę do zarabiania pieniędzy oraz możliwość zwiększenia bądź zmniejszenia wartości zakładanej waluty. Stworzyliśmy własne pikselowe grafiki klasycznych hazardowych symboli.

Zdjęcia nr 5 i 6 przedstawiające końcowy wygląd jednorękiego bandyty

5.2. Elementy niezrealizowane

Pierwotny plan zakładał wykorzystanie buzzera czy też głośniczka, aby odgrywana była melodia gdy gracz wygrywa bądź przegrywa. Pomysł ten nie został zrealizowany z dwóch powodów. Pierwszym z nich jest fakt, że odpowiednie zaprogramowanie buzzera oraz stworzenie melodii zostało uznane za zbyt czasochłonne i nie współmierne do istotności tego elementu. Drugim powodem jest stwierdzenie, że dla użytkownika ciągłe słuchanie melodii przy wygranej bądź przegranej po pewnym czasie zaczęłoby być irytujące, co jest wprost sprzeczne z celem który chcielibyśmy osiągnąć. Z tych powodów element muzyczny został porzucony.

5.3. Możliwy rozwój projektu

5.3.1. Koncept możliwej przyszłości

W bieżącym momencie Jednoręki bandyta będąc w pełni funkcjonalnym urządzeniem jeszcze nie osiągnął pełni swoich możliwości. Dostrzegana jest możliwość rozwoju w bardzo specyficznym, jednak konkretnym kierunku, który zostanie tu poruszony. Zamienienie Jednorękiego bandyty w prawdziwą maszynę do hazardu dostosowaną do realiów współczesnego, jak i przyszłego świata. Obecnie obrót walutami, transakcjami czy umowami odbywa się głównie w formie cyfrowej. Niektóre kraje postulują wysunięte w przyszłość plany pozbycia się waluty w formie gotówki na rzecz płatności kartą, czy też za pomocą NFC.

Zważywszy na ciągle prężny rozwój kryptowalut, który niezależnie od wielkich wzrostów, spadków, afer finansowych czy krachów nieprzerwanie się rozwija i rozrasta. Sugeruje to, że rzeczone kryptowaluty prędzej czy później staną się istotnym elementem codziennego życia każdego człowieka. Zmiany które nadejdą wpłyną na niszowy rynek jakim jest hazard przy użyciu maszyn. Klasyczne maszyny staną się przestarzałe zważywszy na to, że używają gotówki, chociażby do wypłacania nagród. Gdyby jednak unowocześnić je, stworzylibyśmy urządzenie które zamiast wpłat i wypłat gotówki obsługiwałoby transakcje za pomocą przelewów walut bądź kryptowalut. Oznacza to że musiałoby posiadać moduł odpowiedzialny za łączenie się z siecią, prawdopodobnie bezprzewodowo. Jednocześnie urządzenie mogłoby być w swoich wymiarach znacząco mniejsze od urządzeń współczesnych i klasycznych, jednocześnie nie będąc pełnoprawnym komputerem, gdyż nie byłoby to wymagane do funkcjonowania. Wszystkie te koncepcyjne funkcje mógłby zawierać nasz projekt jeżeli byłby w znaczącym stopniu dalej rozwijany, choć wtedy mogłaby wystąpić sytuacja jak ze statkiem Tezeusza.

5.3.2. Wymagane zmiany software i hardware

Aby powyżej opisany koncept rozwoju jednorękiego bandyty mógł zaistnieć wymagane byłyby zmiany na praktycznie każdym poziomie projektu. Obecnie zastosowany Atmega328 zawarty w Arduino Uno posiada dwa problemy, jednym z nich jest brak legalnej możliwości wykorzystania go w celach zarobkowych, czyli również jako bazowy hardware. Drugim problemem będą oczywiście możliwości obliczeń jak i pamięci, które będą za małe aby rozszerzyć Arduino o moduły obsługi wi-fi jak i software'owe moduły obsługi transakcji cyfrowych. Powoduje to, że należałoby wymienić system mikroprocesorowy na inny, który rozwiązałby oba występujące problemy. Następująco należałoby napisać oprogramowanie praktycznie od zera, gdyż wysoce prawdopodobnym jest konieczność zastosowania innego języka niż był użyty do oprogramowania Atmega328. Należy również rozważyć, czy urządzenie to powinno być energetycznie zależne od innego źródła, czy może powinno posiadać własny akumulator pozwalający na czasową autonomiczną pracę.

Bibliografia

1. Dokumentacja do sterownika ST77355
2. Adafruit GFX Graphics Library - Phillip Burgess
3. DISPLAY CUSTOM BITMAP GRAPHICS ON AN ARDUINO TOUCH SCREEN AND OTHER ARDUINO COMPATIBLE DISPLAYS - Nick Koumaris
4. https://javl.github.io/image2cpp/ - 20.01.2023
5. Poradnik producenta - https://www.waveshare.com/wiki/1.8inch_LCD_Module 20.01.2023
6. https://docs.arduino.cc/ 20.01.2023
7. https://stackoverflow.com/ 20.01.2023
8. https://wiki.microduinoinc.com/ 20.01.2023

Załączniki

Do raportu zostały dołączone

- Kod źródłowy projektu wraz z odpowiednimi komentarzami
- Schemat projektu
- Prezentacja multimedialna