DOKUMENTACJA PROJEKTU Z PRZEDMIOTU
SYSTEMY WBUDOWANE

KODOWANIE MORSE'A

Autorzy:
Katarzyna Kalota
Tomasz Kleiner
WFiIS, Informatyka Stosowana, rok III

Opis projektu

W ramach projektu zrealizowano aplikację, która pozwala na wykorzystanie płytki LPC1768 oraz dołączonych do niej peryferiów jako urządzenia kodującego wiadomości w kodzie Morse'a, oraz potrafiącego prezentować ten kod na kilka sposobów - poprzez odtwarzanie dźwięków, wyświetlanie znaków i wysyłanie sygnałów elektrycznych. Może służyć ono zarówno jako element pośredni, jak i końcowy w układzie przesyłania wiadomości. Ponadto, jeśli nadawca i odbiorca znajdują się z dala od siebie, to nadawca wciąż może w łatwy sposób skontrolować poprawność kodowania, gdyż pierwotna wiadomość jest do niego odsyłana w kodzie Morse'a za pomocą kropek i kresek.

Wykorzystane podzespoły

W projekcie zostały wykorzystane następujące podukłady:

- Płytka ewaluacyjna Open 1768
- Wyświetlacz LCD (rozdzielczość: 320x240)
  - sterownik ILI9325
- Moduł interfejsu RS232 (RS232 Board), Waveshare
- Moduł analogowy dla przetworników ADC i DAC (Analog Test Board), Waveshare

Możliwości i założenia projektu

Aplikacja nasłuchuje na interfejsie UART w poszukiwaniu ciągu znaków wprowadzonego przez użytkownika. Po jego otrzymaniu, następuje jego konwersja do alfabetu Morse'a, a jej wynik jest prezentowany jednocześnie na 3 sposoby:

- Poprzez emisję długich i krótkich dźwięków za pomocą głośnika będącego częścią modułu przetwornika DAC,
- Poprzez prezentację zakodowanej wiadomości na wyświetlaczu, w formie kropek (sygnał krótki) i kresek (sygnał długi). Dodatkowo, przetłumaczona wiadomość jest odsyłana do nadawcy przez UART, aby mógł on skontrolować poprawność kodowania,
- Poprzez ustawianie stanu wysokiego jednego z pinów (konkretnie P2.0) odpowiednio do kodowania Morse'a.

Dokładna konfiguracja wyjścia przedstawia się następująco:

- W przypadku emisji dźwięków przez głośnik, sygnał długi (kreska) jest odwzorowany poprzez odegranie ciągłego dźwięku o stałej amplitudzie (można ją ustawić w kodzie programu) przez jedną sekundę. Sygnał krótki (kropka) to odegranie podobnego dźwięku, jednakże na czas o połowę krótszy. Przerwa pomiędzy dwoma sygnałami kodującymi tę samą literę lub cyfrę sygnalizowana jest brakiem dźwięku przez pół sekundy, przerwa pomiędzy ostatnim sygnałem poprzedniej litery lub cyfry a pierwszym sygnałem kolejnej to jedna sekunda ciszy, natomiast przerwa między dwoma słowami (realizowana jako spacja) to brak dźwięku przez 2 sekundy.
- W przypadku ustawiania stanu na pinie wyjściowym, obowiązują te same interwały czasowe, co w punkcie wyżej - natomiast sama realizacja polega na ustawieniu stanu wysokiego pinu wówczas, gdy nadawany byłby sygnał dźwiękowy, oraz stanu niskiego, kiedy panowałaby cisza. Dzięki temu, możliwe byłoby podłączenie kolejnego urządzenia, które mogłoby wczytywać zakodowane dane i poddać je dalszej obróbce, np. dekodowaniu.
- W przypadku wyświetlania zakodowanej kodem Morse'a wiadomości na ekran (oraz odsyłania jej do nadawcy przez UART), sygnał długi prezentowany jest jako kreska, a krótki jako kropka. Pomiędzy sygnałami kodującymi ten sam znak dodana jest niewielka przerwa na wyświetlaczu, pomiędzy ostatnim sygnałem kodującym poprzednią literę a pierwszym kodującym następną pojawia się większa przerwa, natomiast pomiędzy dwoma słowami pojawia się znak '/'.

Ograniczenia

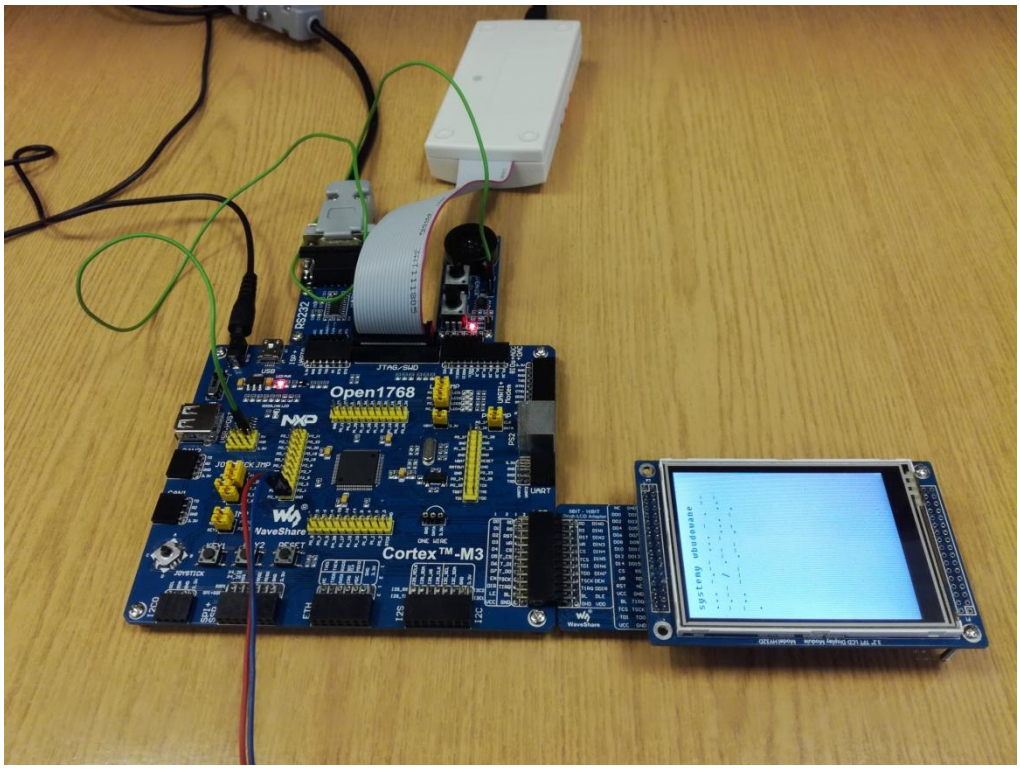
Obecnie w urządzeniu istnieją następujące ograniczenia:

- Tekst może zawierać tylko litery alfabetu angielskiego, cyfry oraz spacje. Jeśli zostanie wprowadzony inny znak, urządzenie przetworzy całą wiadomość, natomiast w miejscu nieznanego znaku na wyświetlaczu pojawi się znak '!', informujący o błędzie w danych wejściowych. Dla owego znaku nie zostanie odtworzony żaden dźwięk, i nie zostanie ustawiony stan wysoki na pinie wyjściowym.

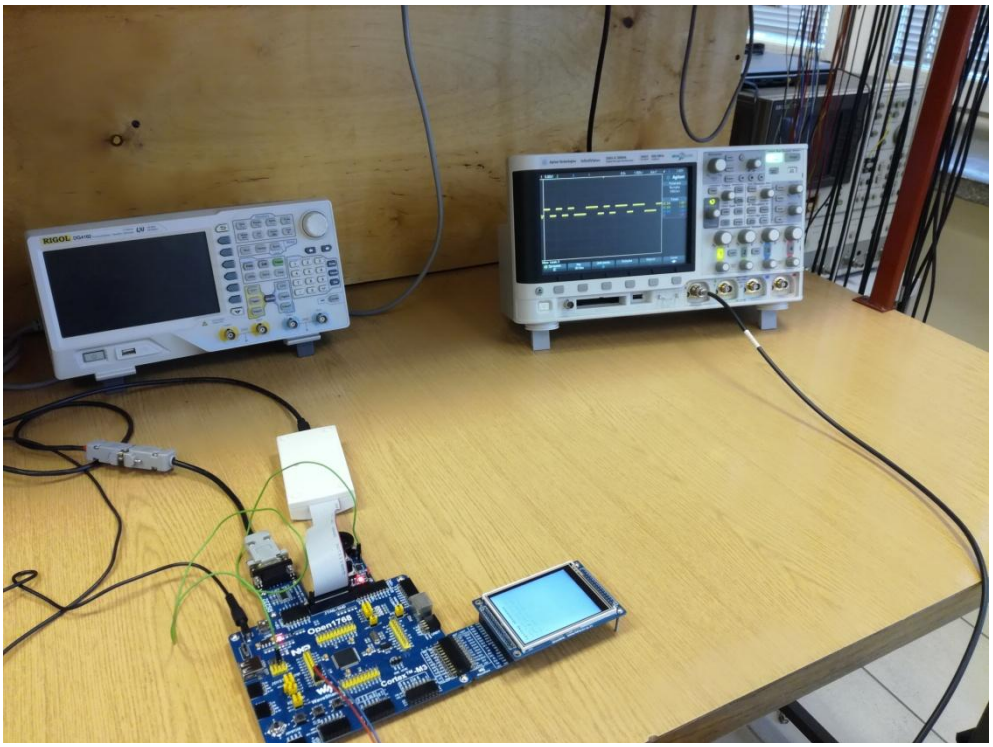
- Jednorazowo można wysłać tekst o długości 50 znaków, wliczając w to spacje. Ograniczenie to jest podyktowane wielkością wyświetlacza - w przypadku dłuższych wiadomości, często nie cały kod Morse'a mieścił się na ekranie, a nie zaimplementowano jego przewijania.
- Program pozwala na wysłanie tylko jednej wiadomości (jej wysłanie odbywa się po odczytaniu znaku nowej linii na UART, bądź też przy wczytaniu 51-ego znaku). Aby wysłać kolejną wiadomość, należy zresetować płytkę, i dopiero wtedy nadać kolejną wiadomość.
- Jeśli użytkownik wprowadza tekst za pomocą terminala UART, np. z komputera PC, to nie jest możliwe poprawienie błędu za pomocą wykasowania danego znaku (np. poprzez naciśnięcie klawisza 'backspace') i wstawienie tam innego znaku. Najprostszym sposobem jest zresetowanie płytki i wprowadzenie danych jeszcze raz, ponieważ maksymalna długość wprowadzanego tekstu jest na tyle mała, że ponowne napisanie wiadomości nie powinno stanowić większego problemu.

Instrukcja obsługi

Po podłączeniu układów oraz kompilacji programu, na UART wpisujemy słowo lub zdanie, które chcemy zakodować. Przesłanie danych następuje po przekroczeniu dozwolonej liczby 50 znaków lub po wciśnięciu klawisza ENTER. Następnie, pod wpisanym tekstem na UART'cie pojawi się jego wersja w kodzie Morse'a. Wprowadzony tekst pojawia się na wyświetlaczu wraz ze swoją zakodowaną wersją (najpierw tekst, pod nim kod Morse'a - każde słowo jest oddzielone znakiem '/', natomiast poszczególne litery - podwójną spacją). W tym samym czasie, głośnik nadaje sygnały dźwiękowe, a na pinie wyjściowym na odpowiednie odstępy czasu pojawiają się stan wysoki (na ok. sekundę dla kreski i pół sekundy dla kropki) oraz niski. Aby wprowadzić nowy tekst, należy nacisnąć przycisk RESET na płytce i wpisać go na UART.



Płytka Open 1768 wraz z podłączonymi podzespołami. Na ekranie wyświetlone są: wprowadzony tekst oraz jego wersja w kodzie Morse'a



Układ podłączony do oscyloskopu, który wizualizuje zmiany sygnału emitowanego przez płytkę poprzez pin wyjściowy, odpowiadające długim i krótkim sygnałom kodu Morse'a

Opisy funkcji wykorzystanych w projekcie:

- ***void UART_init()*** - inicjalizacja UART, konfiguracja pinów i rejestrów,
- ***void sendCharToUART(char c)*** - funkcja czeka na możliwość wysłania znaku na UART, a następnie przesyła przekazany jako argument znak,
- ***void sendStringToUART(char* tab)*** - analogicznie jak funkcja wyżej, jako argument przekazane wiele znaków,
- ***char readCharFromUART()*** - funkcja czeka na znak z UART'a, pobiera go, a następnie go zwraca,
- ***void fillMorseTab()*** - funkcja przy pomocy funkcji *translate* wypełnia globalną tablicę *morseTab* kodami Morse'a dla liter alfabetu angielskiego oraz cyfr,
- ***char* translate(char ch)*** - funkcja otrzymuje jako argument literę lub cyfrę i zwraca jej odpowiednik w kodzie Morse'a,
- ***void readWordToTable()*** - funkcja odczytuje znaki z UART'a , wpisuje je do tablicy *textFromUART* a następnie przy pomocy funkcji *napisNaWyswietlacz* wypisuje na wyświetlaczu wprowadzony z UART'a tekst,
- ***void TIMER1_IRQHandler()*** - Handler odpowiadający za sterowanie dźwiękiem
    - sprawdzanie, czy dźwięk ma zostać odtworzony oraz czy na pinie ma zostać ustawiony stan wysoki:
        - jeśli tak, to dla każdego znaku z tablicy *textFromUART* ustala jak długo ma odtwarzać dźwięk, ustawiając rejestr MR0 na odpowiednią wartość oraz ustawia stan wysoki i wysyła na pin sygnał o odpowiedniej długości,
        - jeśli nie, to ustalamy długość przerwy w odtwarzaniu dźwięku na podstawie pozycji w tablicy (czy ma być to koniec znaku '-' lub '.',

koniec litery/cyfry czy koniec słowa) oraz ustawia stan niski na właściwy okres czasu,

- ***void TIMER0_IRQHandler()***- Handler odpowiadający za generowanie dźwięku w oparciu o tablicę z wartościami sinusa,
- ***void TIM_init()*** - inicjalizacja Timerów 0 i 1, konfiguracja pinów i rejestrów,
- ***void initSinus(int amplitude)*** - funkcja odpowiadająca za generowanie sinusa o odpowiedniej amplitudzie (na jego podstawie generowany jest dźwięk). Minimalna wartość to 1, maksymalna to 512. Obecnie ustawiona jest amplituda równa 2, aby sygnał dźwiękowy był akceptowalnie cichy,
- ***void PinConfigure_forDAC()*** - konfiguracja pinów dla modułu DAC,
- ***void setOutputPin()*** - konfiguracja pinu 2.0, który służy jako wyjście dla sygnału prostokątnego odpowiadającemu kodowi Morse'a,
- ***void screen_config()*** - inicjalizacja ekranu, konfiguracja pinów i rejestrów,
- ***void piszNaWyswietlaczu(char napis,int pozX, int pozY)*** - funkcja wypisująca jeden znak na wyświetlaczu,
- ***void napisNaWyswietlacz(char* word, int length,int startX, int startY)*** - funkcja wyświetlająca przy pomocy funkcji *piszNaWyswietlaczu* napis wprowadzony z UARTa,
- ***void morseNaWyswietlacz(char* word,int length, int startX, int startY)*** - funkcja wyświetlająca przy pomocy funkcji *piszNaWyswietlaczu* kod Morse'a.

Schemat działania kodu:

- inicjalizacja ekranu oraz UART'a
- ustawienie koloru tła na ekranie
- wypełnienie tablicy słownikowej *morseTab*
- konfiguracja pinu wyjściowego
- oczekiwanie na dane wejściowe
- wczytanie z UART'a znaków przeznaczonych do kodowania
- wypisanie na UART zakodowanego tekstu
- wypisanie na ekran tekstu oryginalnego oraz zakodowanego przy jednoczesnym generowaniu dźwięku oraz wysyłaniu sygnałów na pin wyjściowy (w tym miejscu następują przerwaniaTIMER0 oraz TIMER1)