Podstawy Techniki Mikroprocesorowej II

Temat: Robot mobilny ,,Arnold”

Wykonawca:	
Imię i Nazwisko Nr indeksu, wydział	Jakub Kolasa 249012, W4
Prowadzący kurs	mgr inż. Wojciech Tarnawski
Termin zajęć	Pon, 8:00-11:00

1. Cel projektu.

Celem projektu było stworzenie robota mobilnego sterowanego zdalnie. Robot jest złożony z kilku niezależnie działających modułów. Ich praca jest koordynowana poprzez strukturę ,,states" dzięki której robot może wchodzić we wcześniej zaprojektowane stany.

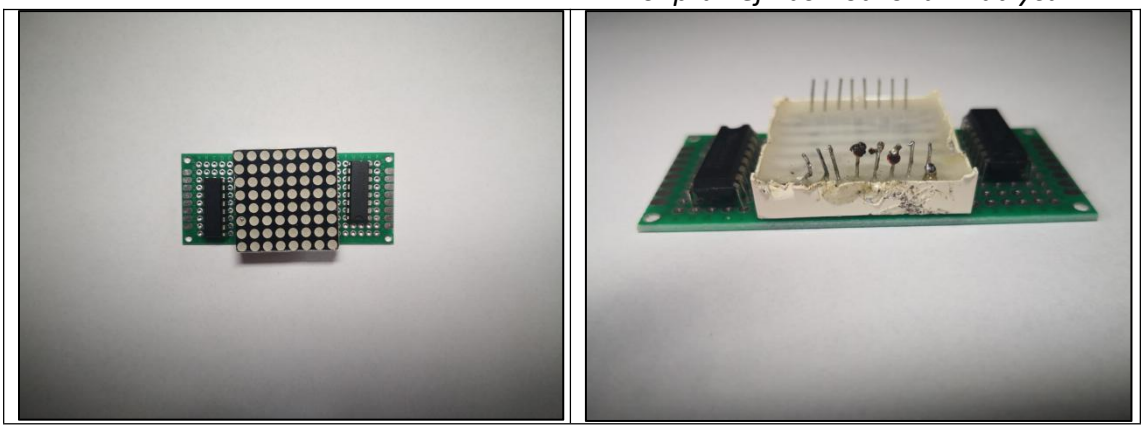
2. Spis części.

Do realizacji projektu wykorzystano:

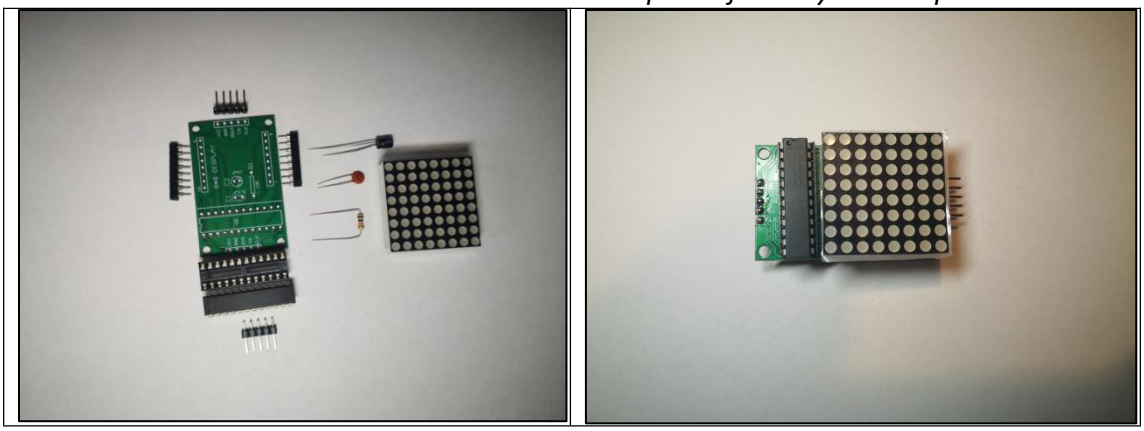
a) Matryca LED 8x8 jako twarz robota (1088bs) sterowana układem MAX7219

Początkowo matryca miała być sterowana przy pomocy dwóch 8-bitowych rejestrów przesuwnych. Jednak podczas rozlutowywania przypadkowo połączonych nóżek uszkodzono matrycę. Niestety kupienie samej nowej matrycy okazało się niemożliwe, dlatego zakupiono matrycę wraz z układem MAX7219.

Po lewej: początkowo planowany układ
Po prawej: uszkodzona matryca LED



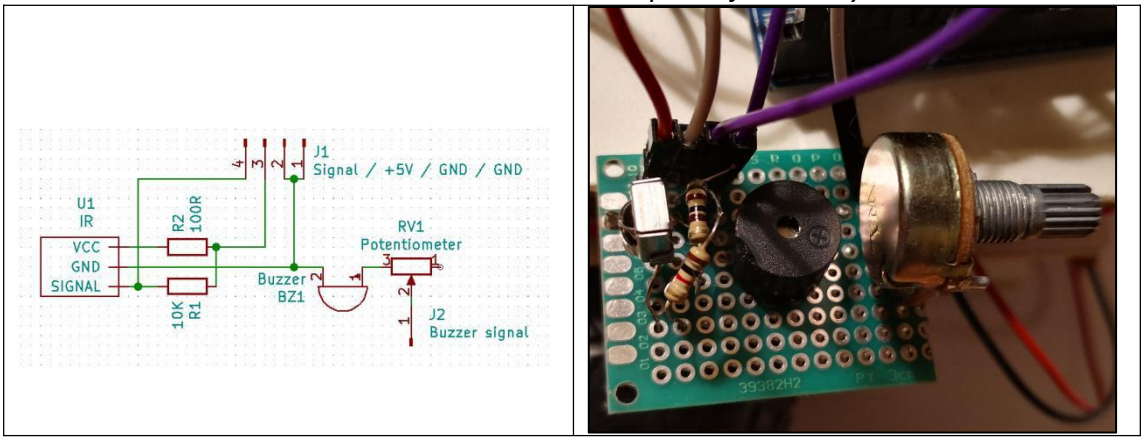
Po lewej: nowy układ z MAX7219 przed zlutowaniem
Po prawej: nowy układ po zlutowaniu



b) Dioda podczerwona do odbioru sygnału sterującego z pilota (VS1838b)
c) Buzzer do sygnalizacji dźwiękowej

Dioda IR, oraz buzzer zostały umieszczone na jednej płytce. Buzzer posiada możliwość manualnej regulacji głośności przy pomocy potencjometru.

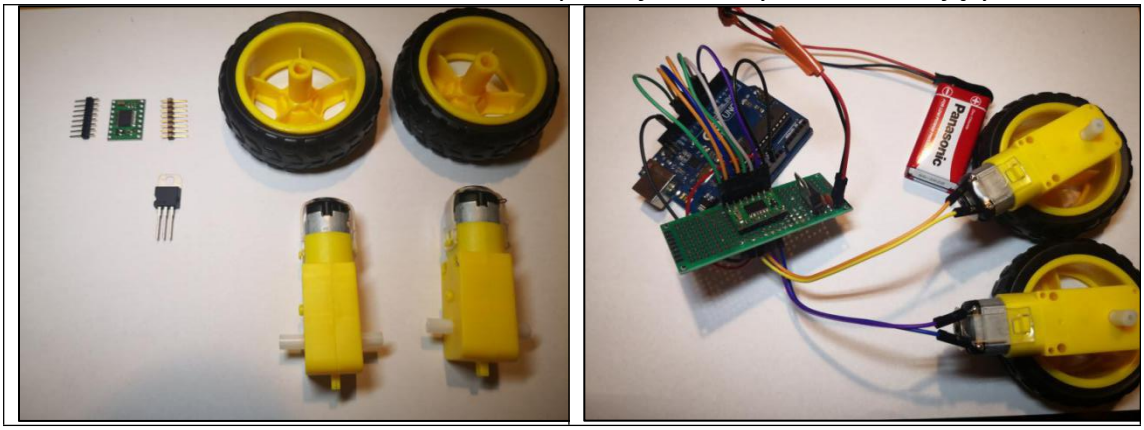
Po lewej: Schemat układ IR + buzzer w programe KiCad
Po prawej: Gotowy układ IR + buzzer



d) Dwa silniki DC z kołami
e) Stabilizator napięcia 5V dla silników
f) Dwukanałowy sterownik silników (TB6612FNG)

Logika układu TB6612FNG jest zasilana z pinów Arduino. Silniki są zasilane przy pomocy stabilizatora napięcia 5V zasilanego baterią 9V.

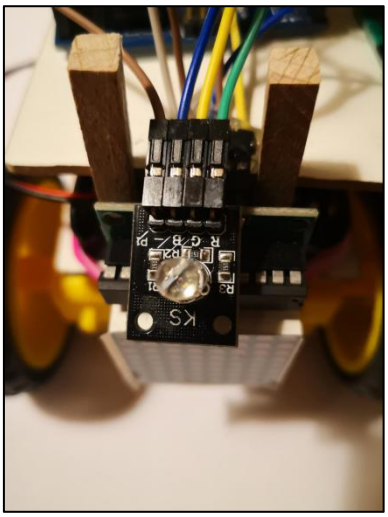
Po lewej: Układ sterujący silnikami przez montażem
Po prawej: Gotowy układ sterujący silnikami



g) Dioda RGB

Moduł RGB może być bezpośrednio wpięty do pinów płytki, ponieważ każda dioda posiada dopasowany rezystor SMD.

Moduł RGB

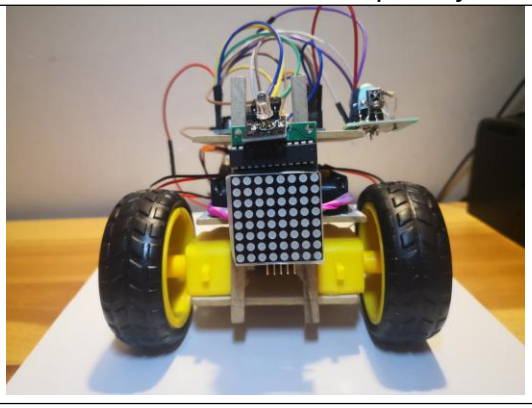


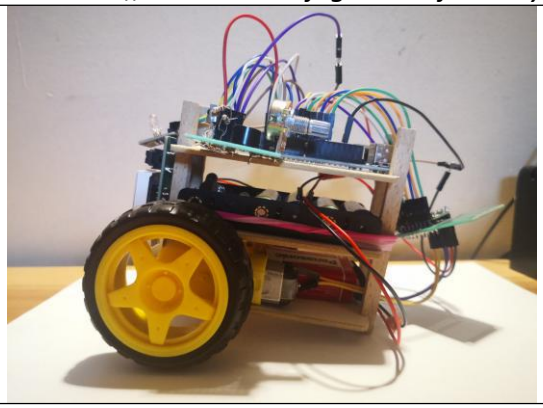
e) Całość sterowana przy pomocy Arduino R3 Uno.

3. Mechanika i rozmieszczenie części.

Ciało robota składa się z trzech płyt (10x7 cm), oraz czterech podpórek (10 cm) z nacięciami w które wchodzą wcześniej wspomniane płytki. W ten sposób otrzymano konstrukcję na kształt regału. Na dolnej półce znajdują się dwa silniki, oraz bateria 9V. Silniki są przypomocowane przy pomocy kleju na gorąco. Na kolejnej półce znajduje się koszyk na 6 baterii 1.5V. Zasilają one Arduino Uno znajdujące się na ostatniej półce. Piny matrycy LED są przymocowane klejem na gorąco do górnej półki. Moduły IR/Buzzer, oraz RGB znajdują się na przodzie górnej półki. Moduł sterujący silnikami znajduje się na tyle robota.

Po lewej: widok robota ,,Arnold" od przodu
Po prawej: widok robota ,,Arnold" od jego lewej strony





3. Program, opis działania poszczególnych modułów.

Każdy z modułów posiada własną reprezentację w postaci struktury. Wszystkie posiadają dwie metody o tych samych nazwach:

a) void getTime(unsigned long milis)

W celu umożliwienia ,,synchroniczności" skorzystano z funkcji biblioteki Arduino millis(). Zwraca ona liczbę typu unsigned long informującą o liczbie milisekund jakie minęły od uruchomienia płytki Arduino. Każda struktura zapamiętuje poprzednio otrzymany czas i na podstawie różnicy z czasem aktualnym stwierdza ile czasu musi poczekać do wykonania kolejnej akcji. W pewien sposób jest to zastępstwo funkcji delay(), która wstrzymałaby działanie calego programu.

b) void setup()

Funkcja inicjalizująca działanie modułu.

a) Matryca LED 8x8 (1088bs) sterowana układem MAX7219 (ledMatrix.h)

Sterowanie matrycą odbywa się poprzez wysyłanie do sterownika liczby opisującej stany na kolejnych 64 diodach matrycy. Na początek należy włączyć sterownik w stan wczytywania poprzez zmiane stanu na pinie CS na niski. Następnie poprzez manipulacje pinami DIN, oraz CLK przesyła się dwa bajty danych. Pierwszy zawiera informacje na temat stanu diod, a kolejny na temat adresu (numeru rzędu na matrycy). Drugi bajt jest w połowie ,,pusty" ponieważ do opisu adresu potrzebuje on jedynie 4 bitów.

Przebiegi pinów sterownika MAX7219

Źródło: https://datasheets.maximintegrated.com/en/ds/MAX7219-MAX7221.pdf

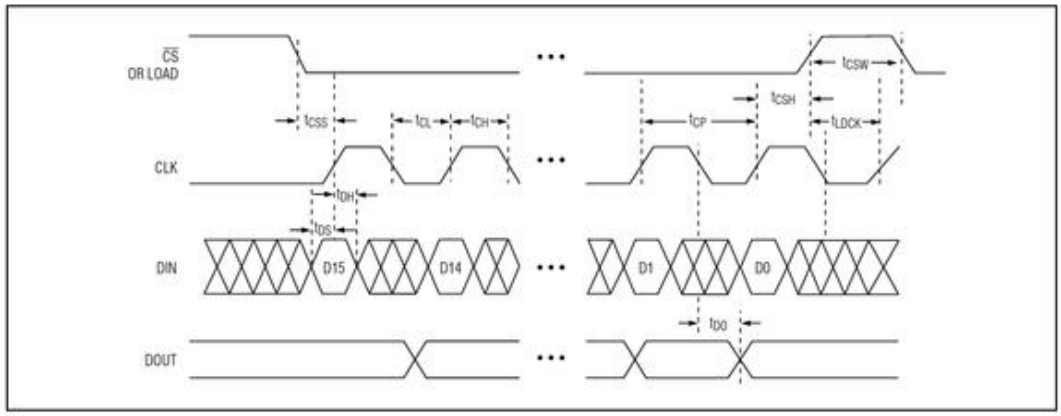


Figure 1. Timing Diagram

Table 1. Serial-Data Format (16 Bits)

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	ADDRESS				MSB	DATA						LSB

Opis metod struktury *ledMatrix*

void changeImageTable(int64_t hex):

Zmiana obrazu wyświetlanego na matrycy. Opisywana metoda dzieli 8-bajtową liczbę na 8 liczb o rozmiarze jednego bajta i zapisuje to do komórek tablicy. Do wytwarzania obrazów wykorzystano aplikacje tworzącą 8-bajtową liczbę HEX dla danej konfiguracji.

Program wykorzystany do tworzenia klatek obrazu

https://xantorohara.github.io/led-matrix-editor/#0000000000000000

void send_byte(unsigned char data):

Wysyła jeden bajt danych do sterownika.

void send_data(unsigned char address,unsigned char dat):

Wysyła dane pod zadany adres. W tym przypadku jest to wysłanie konfiguracji LED do wyświetlenia w zadanej kolumnie.

void display():

Wyświetla obraz zapisany w imageTable.

void init_MAX7219():

Inicjuje działanie sterownika. Metoda zapożyczona ze strony
https://tinusaur.org/tag/max7219/

void play(animation* a):

Rozpoczęcie odgrywania nowej animacji.

void getTime(unsigned long milis):

Zmienia klatki animacji po upłynięciu czasu ich trwania.

W celu wykonywania animacji stworzono dwie pomocnicze struktury.

a) frame - informacja o obrazie (liczba w systemie szesnastkowym, każdy bit odpowiada dziodzie na matrycy), oraz długość trwania danej klatki.

b) animation - wskaźnik do tablicy klatek, oraz liczba klatek w tablicy.

b) Dioda podczerwona (VS1838b) do odbioru sygnału sterującego z pilota (IRdiode.h)

Pilot wysyła do diody ciąg impulsów. Każdemu przyciskowi odpowiada konkretna konfiguracja impulsów. W programie wykorzystano bibliotekę arduino *IRremote.h*, która obsługuje pracę diody IR i odczytuje sygnał w postaci liczby HEX. Dana liczba jest zapisywana w zmiennej *actionNumber,* która jest potem interpretowana w strukturze ,,states".

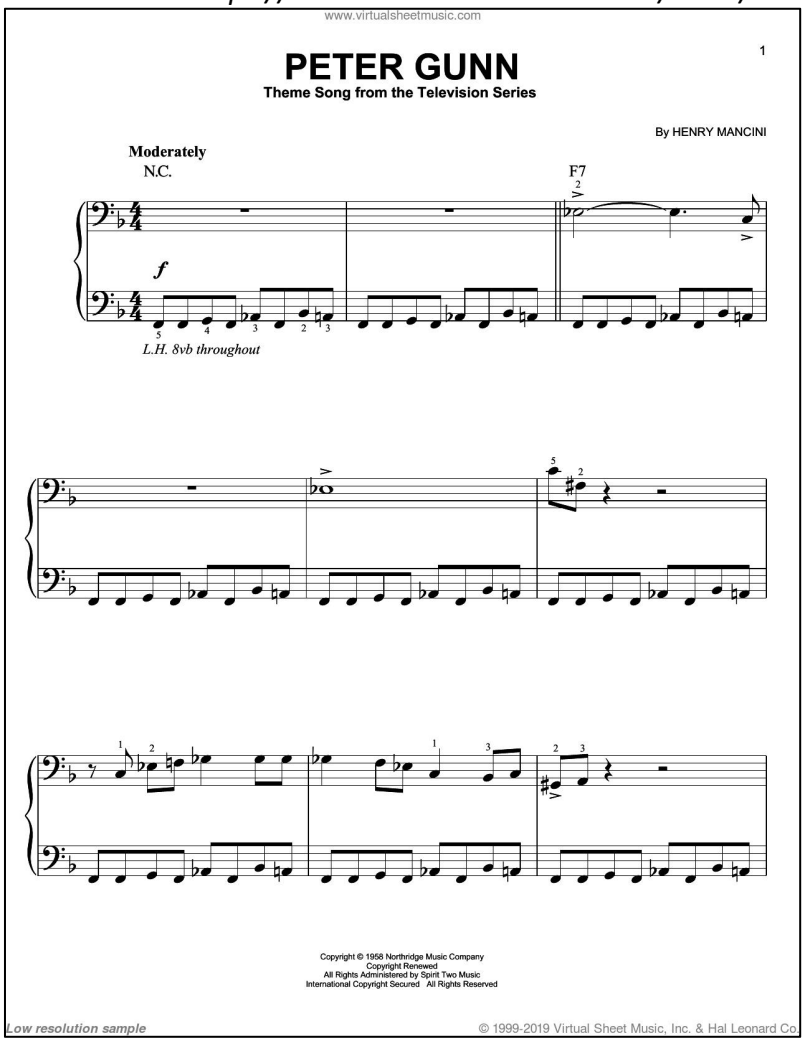
Zastosowany pilot
wraz z liczbami HEX odpowiadającymi odpowiednim przyciskom
Źródło: https://my.oschina.net/u/4355947/blog/4663181

c) Buzzer do sygnalizacji dźwiękowej (buzzer.h + songs_and_melodies.h)

Pozbawiony generatora buzzer otrzymuje sygnały o różnej częstotliwości umożliwiając w ten sposób granie melodii. Do odgrywania dźwięków wykorzystano funkcję *tone*() ze standardowej biblioteki arduino. Wysyła ona na wskazany pin sygnał o zadanej częstotliwości. Pomocniczna struktura *note* reprezentuje dźwięk o konkretnej częstotliowości i długości trwania. Długość dźwięku jest zapisywana w postaci referencji do zmiennej dzięki czemu możlwa jest zmiana tempa w trakcie działania programu. Realizuje ją funkcja changeTempo(int newBPM). Druga pomocnicza struktura *melody* zawiera wskaźnik na tablice nut, oraz informację o jej długości. Melodię można odegrać przy pomocy metody play(melody* m) w strukturze *buzzer*.

Do określenia częstotliwości poszczególnych dźwięków wykorzystano gotową listę ze strony:
https://forbot.pl/blog/kurs-arduino-ii-syrena-alarmowa-mosfet-w-praktyce-id15497.

Nuty wykorzystane do stworzenia melodii blues_brothers_PeterGunn
Źródło: https://www.virtualsheetmusic.com/score/HL-175840.html



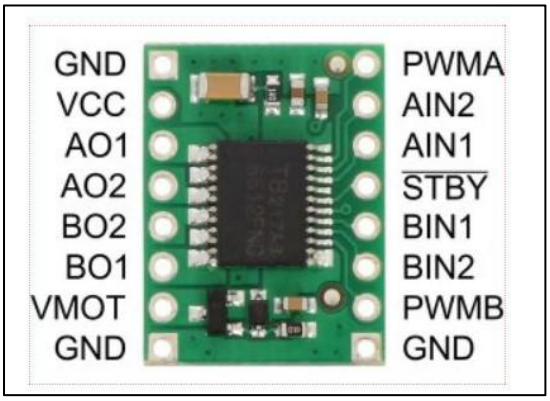
d) Dwukanałowy sterownik silników (TB6612FNG) (hbridge.h)

Do sterowania silnikami wykorzystano sterownik TB6612FNG. Oba silniki są zasilane przy pomocy baterii 9V. Maksymalny prąd jaki może ona wytworzyć jest zbyt mały, aby oba silniki pracowały równocześnie. Dlatego silniki są włączane naprzemiennie w odstępach 100 milisekund. Jest to realizowane przez strukturę hbridge.

Funkcje kontroli mostka H przy pomocy stanów wysoki/niski
Źródło: https://www.sparkfun.com/datasheets/Robotics/TB6612FNG.pdf

Input				Output		
IN1	IN2	PWM	STBY	OUT1	OUT2	Mode
H	H	H/L	H	L	L	Short brake
L	H	H	H	L	H	CCW
		L	H	L	L	Short brake
H	L	H	H	H	L	CW
		L	H	L	L	Short brake
L	L	H	H	OFF (High impedance)		Stop
H/L	H/L	H/L	L	OFF (High impedance)		Standby

Piny układu TB6612FNG
Źródło:
https://botland.com.pl/sterowniki-silnikow-moduly/32-tb6612fng-dwukanalowy-sterownik-silnikow-135v1a-pololu-713.html



e) Dioda RGB (rgb.h)

W zależności od wybranego trybu działania dioda będzie migać w konkretnej konfiguracji zmieniając kolor co 300 milisekund. Tryb można zmienić przy pomocy funkcji changeState(rgb_state newState).

f) states.h

Jest to główna struktura zarządzająca innymi modułami robota. Umożliwia wprowadzanie robota w konkretne wcześniej zaprogramowane stany. Każdy stan ma przypisaną liczbę HEX będącą odpowiednikiem przycisku na pilocie. Przy pomocy zmiennej *actionNumber* modyfikowanej przez strukturę *IRdiode,* oraz metodę *changeState(char _actionNumber)* możliwa jest zmiana stanu robota przy pomocy pilota.

4. Wnioski.

a) Wykorzystanie standardowej baterii 9V uniemożliwia włączenie obu silników na raz ze względu na zbyt mały maksymalny pobór prądu.

b) Platforma w kształcie regału umożliwia swobodny dostęp i prostotę modyfikacji robota.

c) ArduinoIDE sprawiało wiele kłopotów przy importowaniu bibliotek, oraz uniemożliwia jakąkolwiek personalizację. Nie jest to komfortowe środowisko pracy.

5. Załącznik.
Kod programu.

Kod programu znajduje się w repozytorium:

https://github.com/JakubKolasa/PTM-Arnold