Krótka dokumentacja projektu : Inteligentny pilot IR
Autor: Sławomir Tenerowicz

Do wykonania projektu wykorzystano:

- płytkę ArduinoUNO z procesorem ATMega328P,
- odbiornik podczerwieni TSOP1838,
- nadajnik podczerwieni ST187.

W pierwszej kolejności skonfigurowano odbiór sygnału IR. Odbiornik podczerwieni TSOP1838 jest układem scalonym, w którego skład między innymi wchodzi demodulator. W momencie detekcji sygnału zmodulowanego o częstotliwości 38kHz (fizyczne '1' w nadawanym sygnale IR) na wyjściu układu detektora pojawia się stały sygnał '0'. W przypadku braku detekcji sygnału (fizyczne '0' w nadawanym sygnale IR) na wyjściu układu panuje stan wysoki '1'.

Nadajnikami sygnału IR były piloty korzystające z protokołów NEC oraz SAMSUNG. Kodowanie logicznego '0' i '1' w obu przypadkach jest takie samo:

- logiczne '0' dla fizycznego sygnału '10'
- logiczne '1' dla fizycznego sygnału '1000'

Każdy fizyczny sygnał '0' lub '1' trwa przez około 562.5 [us] co daje czas trwania wartości logicznych:

- logiczne '0' : 1.125 [ms]
- logiczne '1' : 2.250 [ms]

NEC korzysta z 4 bajtów niosących informację, nie licząc bitów stopu na końcu sekwencji, które są po prostu fizycznym '0' - brakiem sygnału, oraz 3 bajtów sygnału fizycznego na początku sekwencji jako sygnał startu:

| 3B :: Start | 1B :: Adres | 1B :: zanegowany Adres | 1B :: Komenda | 1B :: zanegowana Komenda|

3 Bajty startu składają się z sekwencji 16 fizycznych '1' – czas trwania 9 ms - po których następuje 8 fizycznych '0' – czas trwania 4.5 ms. W przypadku adresów i komend ma się tu na myśli Bajty logiczne, które ze względu na różną długość trwania logicznego '0' i logicznej '1' skutkują różną długością trwania kolejnych sekwencji.

Ze względu jednak na stosowanie sygnałów zanegowanych, całkowita długość trwania sygnału jest stała. Ilość fizycznie przesłanych bitów wynosi 120 co odpowiada długości sygnału 67.5 [ms].

W przypadku protokołu SAMSUNG sytuacja jest podobna – występują zaledwie dwie różnice:

- sekwencja startowa składa się z 2 Bajtów fizycznych : 8 fizycznych '1' po których następuje 8 fizycznych '0',
- adres jest przesyłany dwukrotnie, bez negacji

| 2B :: Start | 1B :: Adres | 1B :: Adres | 1B :: Komenda | 1B :: zanegowana Komenda|

Informacje te pozwoliły na napisanie funkcji, która odbiera zdemodulowany sygnał IR. Wykorzystano do tego przerwanie zewnętrzne INT0 reagujący na niski poziom sygnału (na wyjściu TSOP1838

panuje stan wysoki, w momencie detekcji sygnału pojawia się stan niski). Wewnątrz funkcji obsługującej przerwanie zaimplementowano dwa opóźnienia _delay_us(), które pozwoliły na odczytanie fizycznej wartości sygnału mniej więcej w połowie czasu jego trwania:

```
_delay_us(T/2);
czytanie wartości na PIND2;
_delay_us(T/2);
```

Gdzie T to czas trwania fizycznego ‘0’ bądź ‘1’ – 562.5 [us].

Odczytane wartości zapisano w globalnej tablicy statycznej. Sygnał ten został zdekodowany korzystając w prostej logiki. Logikę dopasowano do faktu, że faktyczne wartości sygnału zostały odwrócone i tak:

- logiczne ‘0’ to ‘01’
- logiczne ‘1’ to ‘0111’

W pierwszej kolejności wykrywano ilość fizycznych ‘0’ i ‘1’ znajdujących się na początku sekwencji – Bity Startu. Ilość fizycznych ‘0’ pozwalała później na identyfikację protokołu.

Reszta sygnału była dzielona na 4-segmentowe porcje, pozwalające zidentyfikować występowanie logicznego ‘0’ lub ‘1’. Rozpoznano wystąpienie dwóch różnych kolejności zer i jedynek:

- 0 1 0 1 -> pierwsze dwa bity sugerują występowanie logicznego ‘0’, kolejne dwa mogą być logicznym ‘0’ bądź początkiem logicznej ‘1’ – 0111

- 0 1 1 1 -> logiczne ‘1’

Wykonano działanie: (1 ^ element_1) & element_2 & element_3 & element_4 które zwracało:

- ‘1’ w przypadku logicznej ‘1’, w tablicy przesuwano się o 4 pozycje w przód
-‘0’ w przypadku logicznego ‘0’, w tablicy przesuwano się o 2 pozycje w przód

Zdekodowany sygnał skracano do postaci:

- NEC 3 bajty | 1B :: zanegowany Adres | 1B :: Komenda | 1B :: zanegowana Komenda|
- SAMSUNG 4 bajty | 1B :: Adres | 1B :: Adres | 1B :: Komenda | 1B :: zanegowana Komenda|

Korzystając z przesunięcia bitowego w prawo (>>) wyłuskiwano komendę oraz zanegowaną komendę i przy pomocy operatora XOR sprawdzano poprawność odebranego sygnału. Dodatkowo całość negowano:

negacja [XOR (komenda, komenda zanegowana)]

co w rezultacie dawało ‘0’ jeśli sygnał został odebrany poprawnie, lub wartość różną od 0, jeśli wystąpił błąd przy odbieraniu sygnału.

Dodatkowo, sygnał w skróconej postaci w wersji tabelarycznej zamieniono na liczbę przy pomocy bitowego przesunięcia w lewo (<<). Wraz z sprawdzeniem zgodności komendy i jej zanegowanego odpowiednika pozwalało to na szybką weryfikację poprawności otrzymanego sygnału. W celu

przesłania wartości heksadecymalnej do terminala dzielono ją na 8-bitowe części i przesyłano sekwencyjnie.

Do sprawdzania poprawności powyższego kodu wykorzystano bibliotekę ‘Arduino-IRremote-Master’, która pozwoliła na wstępne odczytanie kodów z pilota NEC oraz SAMSUNG.

Do wysyłania otrzymanego sygnału skorzystano z modulacji PWM. W przypadku protokołu Samsunga, fizyczne ‘1’ jest modulowane sygnałem 38kHZ a współczynnik wypełnienia wynosił ¼ co dało następujące wartości:

- okres trwania sygnału około 26 [us]
- czas trwania stanu wysokiego 6.6 [us]

W celu utworzenia PWM wykorzystano Timer1 (Timer0 odpowiedzialny jest za funkcję *delay* , która mogłaby nie działać poprawnie w przypadku wprowadzania zmian w ustawieniach Timer0). MaskęTIMSK1 dobrano na porównywanie wartości w rejestrach OCR1A oraz OCR1B. Wartość rejestru OCR1A ustawiono na wartość odpowiadającą sygnałowi 38 kHZ :

$$OCR1A = \frac{F\_CPU}{f_{PWM} * 2 * prescaler - 1}$$

Jako że nie korzystano z prescalera (ustawiony na wartość 1) wartość OCR1A wyniosła 210. Ustawienie OCR1B na wartość 70 pozwoliło na uzyskanie wypełnienie ¼ (Timer1 liczy od 0 do 210: startując w 0 ustawia wartość wyjścia na stan Wysoki, gdy napotyka wartość 70 ustawia stan na wyjściu na Niski, gdy doliczy do 210, zaczyna liczyć od 0). Jako wyjście ustawiono PIND3, który umożliwia zastosowanie modulacji PWM.

Do transmisji przekazywano tablicę wartości odebranych przez TSOP1838 w stanie surowym. Funkcję zaimplementowano iteracyjnie:

- dla wartości ‘0’ w tablicy, przez 562.5 [us] wysyłano sygnał PWM
- dla wartości ‘1’ w tablicy przez 562.5 [us] nie wysyłano sygnału, stan niski na diodzie IR

Przerwanie sygnału PWM wykonano globalnym wyłączeniem przerwań. Po wykonaniu transmisji dezaktywowano PWM.