POLITECHNIKA WROCŁAWSKA
WYDZIAŁ MECHANICZNY

KIERUNEK: Mechatronika

Mechatronika w Pojazdach Samochodowych

Podsumowanie projektu: Sygnalizacja hamowania awaryjnego samochodu z wykorzystaniem magistrali CAN

Podsumowanie projektu: Sygnalizacja hamowania awaryjnego samochodu z wykorzystaniem magistrali CAN

Autor:
Sławomir Wawrzyniak 188232

WROCŁAW 2016

SPIS TREŚCI

1. WSTĘP – ZAŁOŻENIA PROJEKTOWE I POZIOM REALIZACJI

Układ ostrzegania o awaryjnym hamowaniu wbudowany w samochód osobowy, w założeniach miał pracować następująco:

Cel projektu, podany w początkowej fazie pracy:
Zapoznanie się z działaniem, obsługą, programowaniem oraz implementacją w rzeczywistym układzie, sieci opartej na protokole CAN.
Założenia projektowe:
System składający się z dwóch mikrokontrolerów:
TMS320F2812 (posiadany zestaw uruchomieniowy)
STM32F407G (wbudowany w zestaw uruchomieniowy Discovery - do zakupu ~109 zł)

Dwóch transcieverów CAN, do wyboru np.:
SN65HVD230 (Texas Inst.)
SN65HVD233 (Texas Inst.)
MEP2551 (Microchip) - 4,86 PLN/szt.

Akcelerometru, np:
LIS244AL (St microel.) - 16 zł/szt. lub wbudowanego w płytę uruchomieniową STM Discovery.
Akcelerometr jednoosiowy przeciążenie mierzone od 0,7g do 1,5g

Działanie sytemu:
Akcelerometr umieszczony na ścianie grodziowej silnika, mikrokontroler STM32F4 zbiera dane z akcelerometru. Mikrokontroler TMS320F2812 w bagażniku samochodu, 2 wyjścia mikrokontrolera podłączone do kierunkowskazów. Mikrokontrolery połączone ze sobą siecią CAN. Mikrokontroler 1. przesyła informację o przekroczeniu przeciążenia 0,7g do mikrokontrolera 2., ten zaś uruchamia światła awaryjne, z częstotliwością 3 Hz.

Główne zadanie: opanowanie standardu CAN

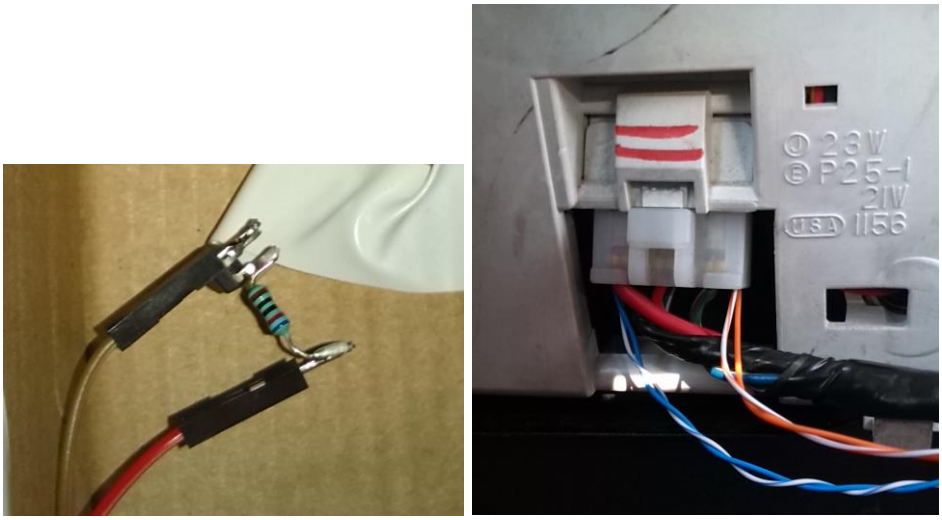
Powyższe założenia udało się zrealizować w całości. Wprowadzono pewne modyfikacje w implementacji końcowej. System w ostatniej fazie rozwoju zbudowany jest w następujący sposób:

System wbudowany został w samochód typu Mazda 626, rok produkcji 1985, wyposażony w silnik o zapłonie iskrowym.

Mikrokontroler STM32F407VGT na płycie STM32F4 Discovery zamontowany jest w bagażniku pojazdu. Tam łączy się z zasilaniem oraz do portu GPIO A1, na jego wejście podpięty jest transoptor, zgodnie z załączonym schematem. Dioda transoptora łączy się z zaciskami żarówki światła stop, co powoduje otrzymanie informacji o wciśnięciu hamulca przez kierowcę (rejestr GPIOA IDR1 zmienia wartość z 1 na 0).

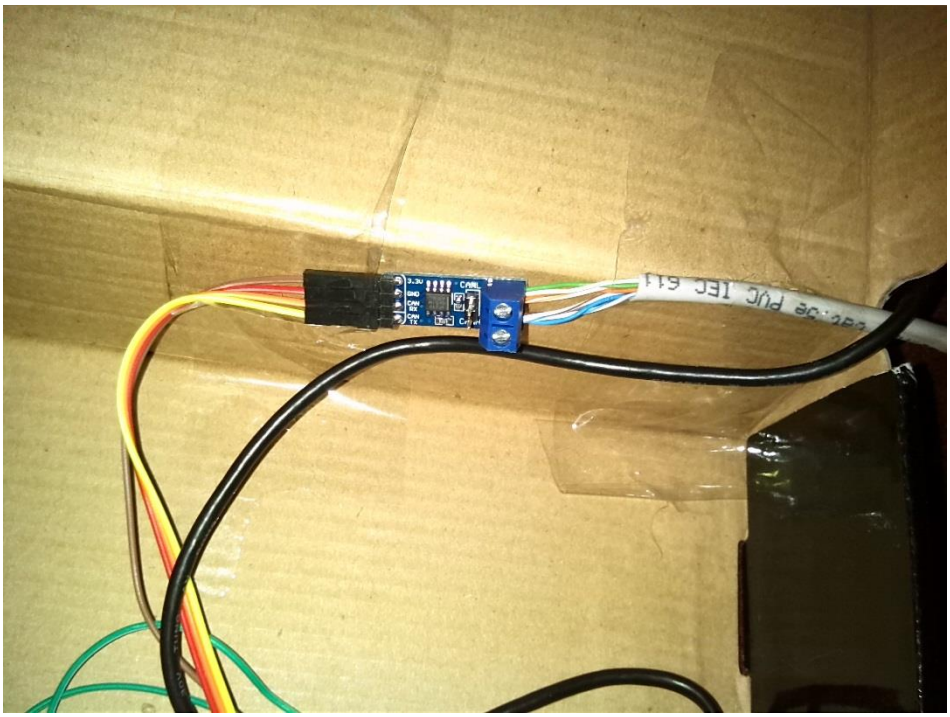
Rysunek 1 STMF4Discovery w bagażniku

Rysunek 2 Zawartość pudełka



Rysunek 3 Transoptor - "pająk", dioda transoptora wpięta w obwód żarówki światła STOP

Do płyty STMF4 Discovery podpięty jest również transciever CAN Waveshare z układem TI SN65HVD230. Przystosowany jest on do układów pracujących na napięciu 3,3V.

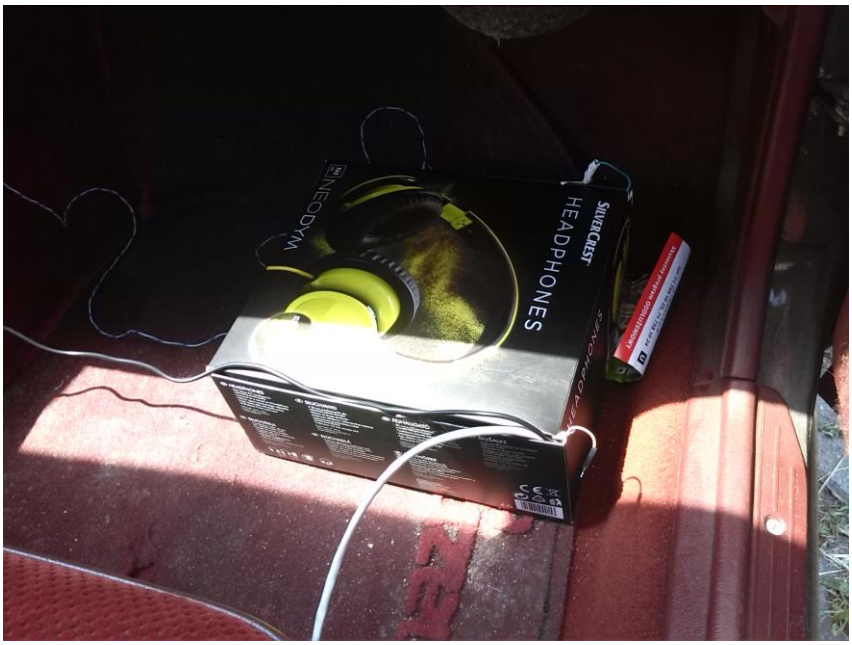


Rysunek 4Transciever CAN na płytce Waveshare

Transciever CAN połączony jest z płytą ewaluacyjną mikrokontrolera TMS320F2812 poprzez skrętkę UTP kat. 5e.

Druga część systemu znajduje się w przedniej części kabiny pojazdu. Płyta ewaluacyjną mikrokontrolera TMS320F2812 została tam podłączona do następujących elementów:
-zasilanie (zasilacz USB 5V)
-przekaźnik sterujący włączaniem świateł – podpięty do portu GPIOB0
-połączenie transcievera na płycie TMS320F2812 z STM32F4 za pomocą skrętki UTP kat 5e.

Rysunek 5 TMS320F2812 z przodu pojazdu



Rysunek 6 Sterownik z przodu pojazdu

2. CZĘŚĆ SPRZĘTOWA


Rysunek 7 Włącznik świateł przeciwmgłowych

Konieczność wpięcia się do instalacji elektrycznej samochodu, wymusiła zmianę założeń projektowych. Na początku planowano uruchamiać światła awaryjne, lecz w danym modelu samochodu dostęp do przekaźnika świateł awaryjnych jest utrudniony. Łatwiejszym sposobem było podłączenie przekaźnika hamowania awaryjnego w miejsce włącznika świateł przeciwmgłowych. Dodatkową zaletą tego rozwiązania jest fakt iż w wiązce dochodzącej do tego włącznika doprowadzono również stałe napięcie 12V służące do zasilania podświetlenia włącznika – a w projekcie posłużyło ono do zasilenia cewki przekaźnika. Styki przekaźnika wpięto w miejsce przeznaczone na styki oryginalnego włącznika. Zatem gdy system zadziała,

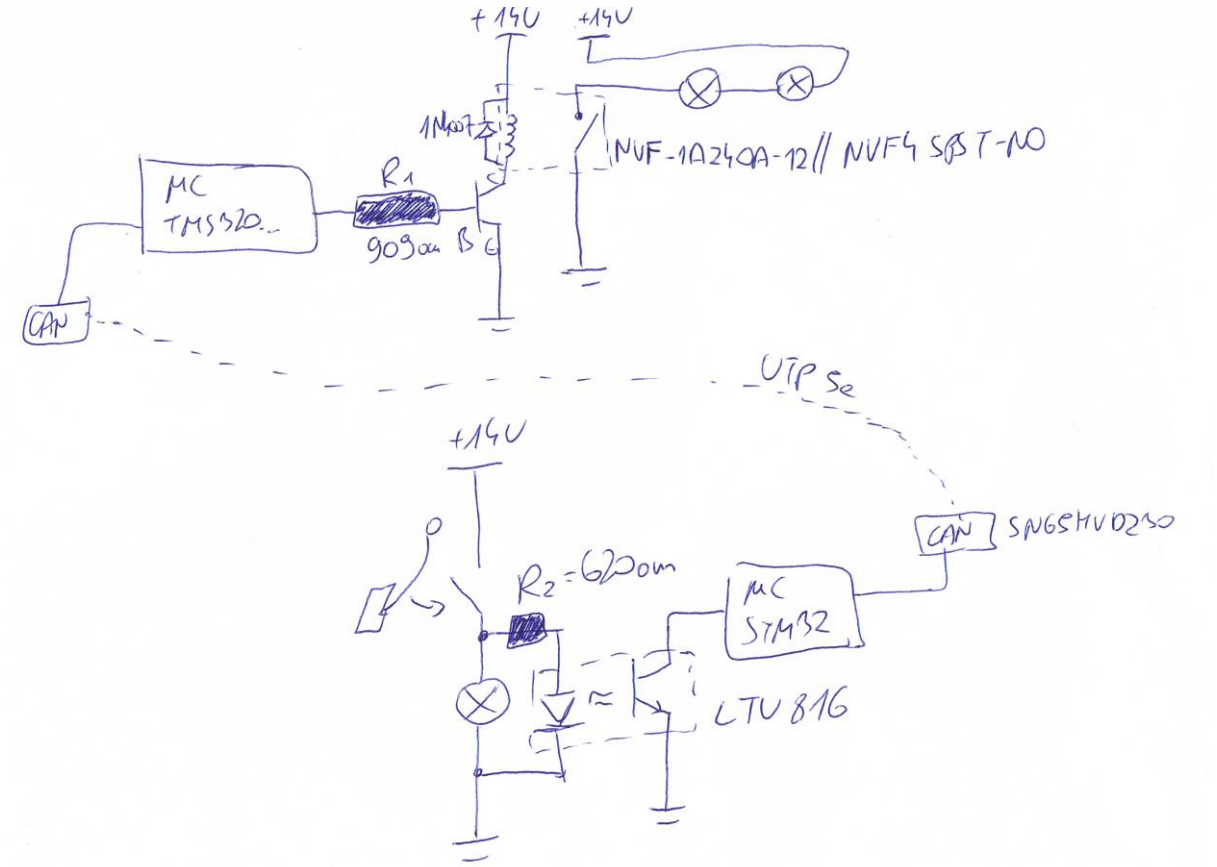
pulsacyjne świecenie dotyczy wyłącznie świateł przeciwmgłowych tylnych.

Rysunek 8Przekaźnik w miejscu włącznika świateł przeciwmgłowych

W projekcie wykorzystano następujące elementy elektroniczne:
-płytę ewaluacyjną STM32F4 Discovery,
-płytę ewaluacyjną z procesorem TMS320F2812,
-2x transciever CAN TI SN65HVD230,
-akcelerometr LIS3DSH firmy ST,
-przekaźnik samochodowy NVF4 SPST-NO, 12V, 40A,
-dioda 1N4007 do zabezpieczenia przeciwprzepięciowego tranzystora sterującego przekaźnikiem,
-transoptor LTV 816,
-rezystor 620 om – ograniczenie prądowe dla diody transoptora,
-rezystor 909 om – ograniczenie prądowe bazy tranzystora,
-2x zasilacz USB 5V.
-przewód UTP kat. 5e
W projekcie nie wykonywano płytki PCB, elementy powiązane z przekaźnikiem przylutowano do niego. Do transoptora przylutowano rezystor oraz wyprowadzenia umożliwiające wpięcie go do listwy goldpin występującej w STMF4Discovery.

Rysunek 9 Załączane światło przeciwmgłowe - analogicznie strona prawa

Schematy elektryczne:



Rysunek 10 Schematy elektryczne

3. CZĘŚĆ PROGRAMOWA

W części programowej skupiono się na kwestii uruchomienia transmisji CAN w obydwu procesorach oraz zaimplementowaniu obsługi akcelerometru LIS3DSH.

Dla procesora TMS320F2812 wykorzystano środowisko pracy Code Composer Studio v 5.5.0, wraz ze sprzętowym emulatorem JTAG - XDS100, co pozwala na podgląd pracy procesora w czasie rzeczywistym.

Dla procesora STM32F407VGT wykorzystano środowisko Keil uVision w wersji 5.18a. Podgląd pracy procesora w czasie rzeczywistym umożliwia wbudowany w płytkę ewaluacyjną ST-LINK/V2 Debugger.

TMS320F2812

Rozpoczynając od środowiska procesora firmy Texas Instruments, przeprowadzono następujące kroki do zaprogramowania procesora TMS320F2812 na odbiór wiadomości (włącz/wyłącz światła):

-załadowano bibliotekę ECan.c do projektu – niezbędną do uruchomienia kontrolera CAN, w bibliotece tej zmieniono parametry rejestru CANBTC, tak by zmienić prędkość pracy magistrali z 1Mbps na 500kbps.

-w programie głównym przeprowadzono żmudna inicjalizację CAN, poprzez ustawienie pożądanych masek akceptacji wiadomości, ustalenia długości identyfikatora (11, 29 bit), włączenie filtrowania wiadomości, ustawienia kierunku działania skrzynek odbiorczych, ustawienia ochrony nadpisywania wiadomości w skrzynkach, włączenie żądanych skrzynek, uruchomienie zalecanych przerwań od błędów i akcji – wszystkie te operacje dla skrzynki 0, 1 oraz 2.

-w pętli głównej programu głównego wykonywana jest następująca procedura:

*jeśli w skrzynce 0 odebrano wiadomość (wiadomość o najwyższym priorytecie zawsze wpada do skrzynki 0) – resetujemy skrzynkę poprzez wpisanie 1 w rejestrze „ECanaRegs.CANRMP.bit.RMP0” i uruchamiamy procedurę migania światłami z częstotliwością 3Hz

* jeśli w skrzynce 1 odebrano wiadomość (wiadomość o niższym priorytecie niż wiadomość dla skrzynki 0) – resetujemy skrzynkę poprzez wpisanie 1 w rejestrze „ECanaRegs.CANRMP.bit.RMP0” – procedura migania światłami zostaje zakończona.

```
/* Configure bit timing parameters */

    ECanaRegs.CANMC.bit.CCR = 1 ;            // Set CCR = 1

    while(ECanaRegs.CANES.bit.CCE != 1 ) {}   // Wait for CCE bit to be set..

    ECanaRegs.CANBTC.bit.BRPREG = 19; //zmiana z 9 na 19 (1mbps na 500kbitps)
    ECanaRegs.CANBTC.bit.TSEG2REG = 2;
    ECanaRegs.CANBTC.bit.TSEG1REG = 10;

    ECanaRegs.CANMC.bit.CCR = 0 ;            // Set CCR = 0
    while(ECanaRegs.CANES.bit.CCE == !0 ) {}   // Wait for CCE bit to be cleared..

/* Disable all Mailboxes  */

    ECanaRegs.CANME.all = 0;        // Required before writing the MSGIDs

}

/*****************************************************/
/* Bit configuration parameters for 150 MHz SYSCLKOUT*/
/*****************************************************/
/*
```

Rysunek 11 Zmiana częstotliwości pracy magistrali

```
    GpioDataRegs.GPBDAT.bit.GPIOB0 = 0;
    for(;;)
    {
        asm ("NOP");

        if(ECanaRegs.CANRMP.bit.RMP0 == 1 ) { //jesli skrzynka 0 otrzyma wiadomosc (a otrzyma wiado
            ECanaRegs.CANRMP.bit.RMP0=1;
                migaj3hz();
            }


        if(ECanaRegs.CANRMP.bit.RMP1 == 1 ) { // jesli przychodzi wiadomosc o wyzszym priorytecie w


                        ECanaRegs.CANRMP.bit.RMP1=1;
                        //wylacz(); //funkcja zawarta obecnie w migaj3hz
        }
```

Rysunek 12 Procedura odbioru wiadomości

```
void migaj3hz(void){
    CpuTimer0.InterruptCount=0;
    while(ECanaRegs.CANRMP.bit.RMP1!=1){ //warunek - jesli przyjdzie wiadomosc o id "wylacz" - przestan migac
            GpioDataRegs.GPBDAT.bit.GPIOB0 = 1;
             while(CpuTimer0.InterruptCount < 2); //czekaj 100 ms
                 CpuTimer0.InterruptCount=0;

            GpioDataRegs.GPBDAT.bit.GPIOB0 = 0;
            while(CpuTimer0.InterruptCount < 2); //czekaj 100ms
                            CpuTimer0.InterruptCount=0;
    }
```

Rysunek 13 Procedura migania światłami

Jak widać w powyższym algorytmie nie interesuje nas zawartość wiadomości, skupiamy się jedynie na fakcie otrzymania wiadomości o konkretnej wartości identyfikatora.

Problematyczna w przypadku tego procesora była ręczna inicjalizacja skrzynek CAN.

Dzięki temu jednak programista może bezpośrednio dowiedzieć się jakie możliwości ustawień są dopuszczalne. Dokumentacja konfiguracyjna jest obszerna, konieczne jest przejście przez każdy krok ze zrozumieniem. Poprawne jej wykonanie pozwala cieszyć się działającym programem. Wykorzystano dokument SPRU074F „TMS320x281x Enhanced Controller Area Network (eCAN) Reference Guide".

STM32F407VGT

Przechodząc do środowiska Keil uVision, aby zaprogramować procesor STM32F407VGT, wykonano następujące kroki:

-załadowano bibliotekę CAN.c zawierającą konfigurację i inicjalizację kontrolera CAN. W tym przypadku nie skupiano się na ustawieniach skrzynek, gdyż powyższy procesor jedynie wysyła informację. Konieczne jest jedynie poznanie mechanizmu wysyłania wiadomości oraz ustawienie odpowiedniej prędkości transmisji danych

-załadowano bibliotekę inicjalizującą transmisję SPI – konieczną do transmisji danych z czujnika przyspieszenia LIS3DSH

```
75    pCAN->MCR = (CAN_MCR_INRQ   |          /* initialisation request          */
76                 CAN_MCR_NART    );        /* no automatic retransmission     */
77                                           /* only FIFO 0, tx mailbox 0 used! */
78    while (!(pCAN->MSR & CAN_MCR_INRQ));
79
80    pCAN->IER = (CAN_IER_FMPIE0 |          /* enable FIFO 0 msg pending IRQ   */
81                 CAN_IER_TMEIE    );       /* enable Transmit mbx empty IRQ   */
82
83    /* Note: this calculations fit for CAN (APB1) clock = 42MHz */
84    brp  = (42000000 / 7) / 500000;        /* baudrate is set to 500k bit/s   */
85
86    /* set BTR register so that sample point is at about 71% bit time from bit start */
87    /* TSEG1 = 4, TSEG2 = 2, SJW = 3 => 1 CAN bit = 7 TQ, sample at 71%      */
88    pCAN->BTR &= ~(((        0x03) << 24) | ((        0x07) << 20) | ((        0x0F) << 16) | (          0x3FF));
89    pCAN->BTR |=  ((((3-1) & 0x03) << 24) | (((2-1) & 0x07) << 20) | (((4-1) & 0x0F) << 16) | ((brp-1) & 0x3FF));
90  }
```

Rysunek 14 Część biblioteki CAN.c odpowiedzialna za ustawienie prędkości działania magistrali

W programie głównym oprócz inicjalizacji procesora oraz kontrolera CAN i SPI, zaprogramowano procedury wysyłania wiadomości.

W pętli głównej programu wykonywane są następujące czynności:
-nawiązanie połączenia z akcelerometrem,
-ustawienie częstotliwości próbkowania akcelerometru oraz aktywnych osi (X, Y, Z)
-pobranie wartości przyspieszenia w żądanej osi

-jeśli przyspieszenie jest większe niż 0,67g, ale mniejsze niż 1,5g oraz jednocześnie wciśnięty jest pedał hamulca – wykonanie procedury włączającej miganie;

-jeśli warunki są niespełnione – wykonanie procedury wyłączającej miganie.

```
197   if((x>0xFFA7 && x<0xFFE0) && przyciskON()==1) {GPIOD->ODR |= 1<<15;
198         //hamowanie ciagle
199     Delay(10);
200     if((x>0xFFA7 && x<0xFFE0) && przyciskON()==1){
201         wlacz3hz();
202         do{
203           GPIOD->ODR |= 1<<15;
204           wlacz3hz();
205         }
206         while ((przyciskON()==1));
207       }
208
209         //Delay(3000); //migaj 3 sekundy
210       }
211       else {GPIOD->ODR &= ~(1<<15);
212         wylacz();
213         }
214
215   }
216 }
```

Rysunek 15 Warunki załączenia ostrzegania o gwałtownym hamowaniu

```
260  void wlacz3hz(void)
261 {//if(g>0,7&&hamulec == 1 && v>>50km/h)
262   int i;
263   CAN_TxMsg[1].id = 0x00000028;                          /* initialize msg to send  tu wpisujemy id wiadomosci
264   for (i = 0; i < 8; i++) CAN_TxMsg[1].data[i] = 0;
265   CAN_TxMsg[1].len = 8;
266   CAN_TxMsg[1].format = EXTENDED_FORMAT;
267   CAN_TxMsg[1].type = DATA_FRAME;
268     if (CAN_TxRdy[1]) {                          /* tx msg on CAN Ctrl #2    */
269   CAN_TxRdy[1] = 0;
270
271   CAN_wrMsg (2, &CAN_TxMsg[1]);
272 }
273 }
```

Rysunek 16 Funkcja włączająca miganie

```
283  void wylacz(void)
284 {//if(hamowanie bylo==1 && czaspohamowaniu==1 && hamulec !=0)
285   int i;
286   CAN_TxMsg[1].id = 0x00000077;                              /* initialize msg to send
287   for (i = 0; i < 8; i++) CAN_TxMsg[1].data[i] = 0;
288   CAN_TxMsg[1].len = 8;
289   CAN_TxMsg[1].format = EXTENDED_FORMAT;
290   CAN_TxMsg[1].type = DATA_FRAME;
291   if (CAN_TxRdy[1]) {                              /* tx msg on CAN Ctrl #2    */
292   CAN_TxRdy[1] = 0;
293
294   CAN_wrMsg (2, &CAN_TxMsg[1]);
295 }
296
297 }
```

Rysunek 17Funkcja wyłączająca miganie

Jak widać na rys. 6 i 7, procedura „włącz” wysyła wiadomość o identyfikatorze 0x28, podczas gdy wyłącz o identyfikatorze 0x77. Wiadomość z ID o mniejszej wartości odbierana jest przez skrzynkę 0, o większej przez skrzynkę 1.

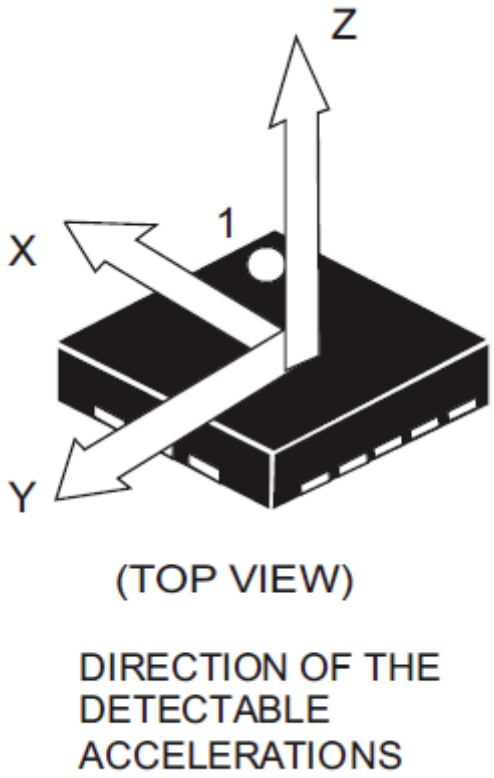
Obsługa akcelerometru LIS3DSH:

-aby prawidłowo odczytać dane z powyższego akcelerometru, należy zapoznać się z jego notą katalogową. Najważniejszymi informacjami, które należy z niej wykorzystać to:

-kierunki wykrywanych przez akcelerometr przyspieszeń

-sposób odczytu i zapisu do i z rejestrów akcelerometru

-funkcjonalności poszczególnych rejestrów.



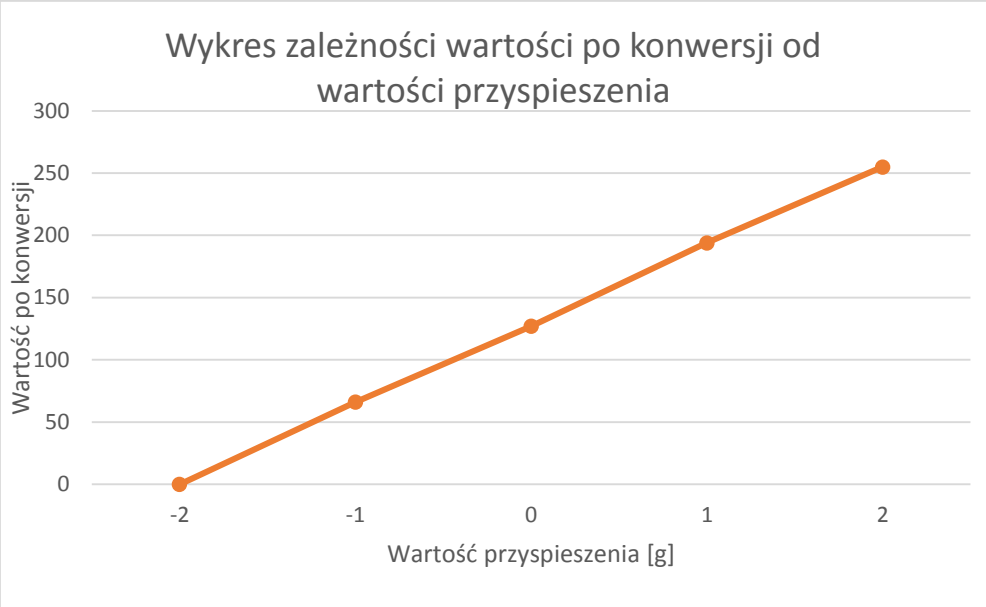
Rysunek 18 Kierunki wykrywanych przyspieszeń

Korzystano głównie z poniższych rejestrów akcelerometru:

CTRL_REG4 – adres 0h23 – odpowiada za ustawienie aktywnych osi akcelerometru (X, Y, Z) oraz częstotliwość próbkowania akcelerometru (od 3,125 Hz do 16kHz).
OUT_X_L – adres 0h29 – wartość przyspieszenia w osi X – bity starsze.

Aby zapisać dane w rejestrach akcelerometru, należy wysłać przez protokół SPI ramkę danych o wartości: 00[6-bitowy adres rejestru] 0000 0000.
Aby odczytać dane z rejestrów akcelerometru, należy wysłać przez protokół SPI ramkę danych o wartości: 00[6-bitowy adres rejestru] 0000 0000.

Ważnym krokiem w projekcje, było wyznaczenie wartości wskazywanych przez akcelerometr, w powiązaniu z wartością przyspieszenia. Na podstawie przyspieszenia ziemskiego oraz zakresu wskazań akcelerometru (ustawiony na +-2g), wykreślono poniższą charakterystykę.



W programie wpisano wartość progową przyspieszenia równą 0,606g. Ograniczono również górną wartość wykrywanego przyspieszenia do mniejszej niż zezwala na to zakres, aby uchronić się przed niechcianymi szpilkami.

Dzięki powyższym implementacjom, udało się osiągnąć zamierzony cel, zaprezentowany na końcowym filmie.

4. Część Mechaniczna

Mazda 626 GC, w której wbudowano system jest już pojazdem wiekowym. Nagrywanie filmu – kilkunastokrotne awaryjne hamowanie z prędkości około 70 km/h do 0, niestety zaowocowało usterką lewego zacisku hamulcowego. Tarcza hamulcowa uzyskała ciemnoniebieską barwę. W dodatku pierwsze nagrywanie odbywało się rejestratorem drogowym – nagrywającym w pętli, o czym zapomniałem – niestety nadpisał wysiłki, jakość obrazu też nie powalała. Za drugim podejściem użyto telefonu komórkowego.
Zaobserwowano, iż na oponach całorocznych Nokian Allweather +, nie jesteśmy w stanie osiągnąć opóźnienia większego niż 0,7g. Powyżej tej wartości na suchym asfalcie blokujemy koła – następuje utrata przyczepności.

5. Prezentacja Wideo

Film prezentujący działanie systemu podczas jazdy umieszczono w repozytorium oraz w serwisie Vimeo, pod adresem:
https://vimeo.com/172238096

Porównując z dostępnym na YT filmikiem prezentującym system ESS, to nie jest najgorzej.
https://www.youtube.com/watch?v=1TmhXd0LrdI