Oprogramowanie dla STM32

Ultradźwiękowy czujnik odległości HC-SR04

Od kilku lat są dostępne w handlu ultradźwiękowe czujniki odległości (sonary) przeznaczone do zastosowań w konstrukcjach amatorskich – robotach i automatyce. Prawdopodobnie najpopularniejszym i najtańszym z nich jest moduł HC-SR04, dostępny w wielu popularnych sklepach internetowych. W artykule przedstawiono moduł czujnika oraz jego współpracę z mikrokontrolerem rodziny STM32.

Pokazany na **fotografii 1** czujnik ultradźwiękowy HC-SR04 zawiera 8-bitowy mikrokontroler Elan EM78P153S oraz wzmacniacze sygnału akustycznego – wyjściowy i wejściowy. Wzmacniacz wyjściowy służy do wysterowania nadajnika ultradźwięków (miniaturowego głośnika). Ciekawostką jest, że do jego zbudowania użyto układu MAX232, znanego z zupełnie innych zastosowań. Wzmacniacz wejściowy (mikrofonowy) został zrealizowany przy użyciu układu LM324 lub podobnego poczwórnego wzmacniacza operacyjnego. Moduł wyposażony jest w jednorzędowe złącze 4-stykowe – wyprowadzenia modułu opisano w **tabeli 1**.

Poziomy wejściowe sygnałów są zgodne ze standardem logicznym TTL, dzięki czemu możliwa jest współpraca modułu z układami zasilanymi napięciem 5 V lub 3,3 V. Poziomem nieaktywnym obu sygnałów jest poziom niski. W celu wykonania pomiaru odległości należy na wejście Trig podać impuls o poziomie wysokim i czasie trwania nie krótszym od 10 µs. Mikrokontroler modułu emituje wtedy paczkę 8 impulsów ultradźwiękowych o częstotliwości 40 kHz i ustawia linię Echo. Jest ona zerowana po odebraniu przez mikrokontroler modułu HC-SR04 dźwięku odbitego od przeszkody. Czas trwania stanu wysokiego na linii Echo określa czas przemieszczania się dźwięku z modułu do przeszkody i z powrotem. Mierząc ten czas, możemy określić odległość pomiędzy czujnikiem i przeszkodą; przy założeniu, że prędkość dźwięku wynosi około 340 m/s, odległość od przeszkody możemy wyznaczyć z wzoru:

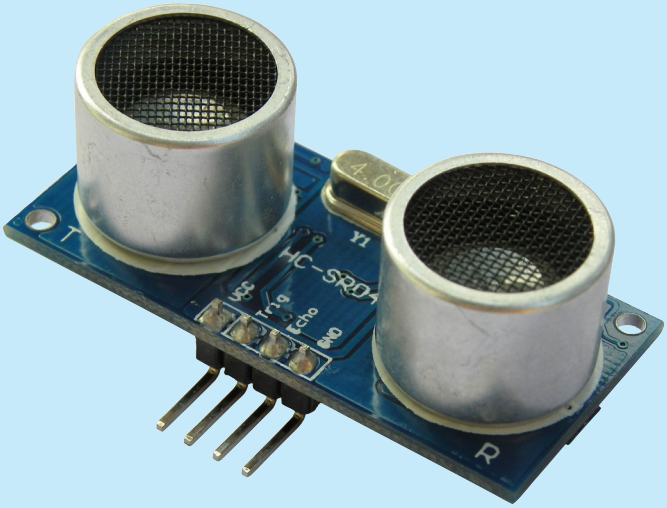
$$d[cm] = \frac{t_{Echo}[\mu s]}{58}$$

Jeżeli moduł nie zarejestruje odbitego dźwięku, linia Echo pozostaje ustawiona i moduł przestaje reagować na kolejne impulsy na linii Trig. W celu ponownego uaktywnienia modułu należy wymusić poziom niski na linii Echo. Ta ważna informacja jest często pomijana w opisach działania modułu.

W praktyce zasięg czujnika odległości nie przekracza 3...5 metrów. Typowo używamy go do pomiaru odległości w zakresie od kilku centymetrów do 2 metrów. Minimalna odległość mierzona wynika z geometrii samego czujnika i wynosi około 2 cm.

Współpraca czujnika z mikrokontrolerem

Typowy scenariusz użycia czujnika odległości zakłada wykonywanie pomiarów odległości ze stałą częstotliwością wynoszącą od 1 do 20 Hz. W celu wykonania pomiaru należy wygenerować impuls na wejściu Trig czujnika i zmierzyć czas trwania impulsu na wyjściu Echo. Aby zagwarantować zakończenie pomiaru, należy również zapewnić kasowanie impulsu na wyjściu Trig poprzez wymuszenie stanu niskiego na linii Trig przy użyciu wyjścia z otwartym drenem. Z doświadczeń



Fotografia 1. Czujnik ultradźwiękowy HC-SR04

Tabela 1. Wyprowadzenia modułu HC-SR04

Nazwa	Opis
Vcc	Zasilanie +5 V
Trig	Wejście impulsu wyzwalającego pomiar
Echo	Wyjście impulsu odwzorowującego czas propagacji dźwięku i wejście wymuszenia końca pomiaru – linia typu „otwarty dren” z rezystorem podciągającym
GND	Masa zasilania i sygnałów

wynika, że czas trwania impulsu kasującego powinien wynosić nie mniej niż 30 µs. Oba impulsy – wyzwalający pomiar i kasujący – mogą być wytwarzane przez dwa wyjścia tego samego timera pracującego w trybie PWM. Do pomiaru czasu trwania impulsu na wyjściu Echo należy użyć kanału timera pracującego w trybie capture, zaprogramowanego na pomiar czasu trwania impulsu lub na chwytanie obu zboczy sygnału wejściowego.

Mikrokontrolery STM32 są wyposażone w timery 4-kanałowe, których poszczególne kanały mogą pracować w różnych trybach. Dzięki temu do obsługi czujnika odległości można użyć jednego timera. Co więcej, pojedyncza linia timera może być równocześnie używana w roli wyjścia PWM i wejście *Capture*, co umożliwia przyłączenie czujnika przy użyciu tylko dwóch wyprowadzeń mikrokontrolera. Do obsługi HC-SR04 można użyć dowolnego z 4-kanałowych timerów TIM1…TIM5. Jedna para kanałów (CH1-CH2 albo CH3-CH4) służy do współpracy z linią Echo. Jeden kanał z drugiej pary jest używany do generowania impulsu wyzwalającego na linii Trig.

Opisany dalej przykład zrealizowano na płytce STM32 Nucleo z mikrokontrolerem STM32F411. Może on zostać łatwo zaadaptowany poprzez modyfikację kodu inicjującego peryferie dla dowolnego mikrokontrolera rodziny STM32F i dowolnego timera 4-kanałowego.

Program przykładowy

Program demonstracyjny wykonuje pomiary odległości ze stałą częstotliwością i przesyła ich wyniki przez port szeregowy. Działa on na płytce Nucleo-F411RE, wyposażonej w interfejs ST-Link/V2-1. Interfejs USART2 mikrokontrolera jest połączony na płytce Nucleo z interfejsem VCOM ST-Link, który jest widoczny w komputerze jako wirtualny port szeregowy. Wyniki pomiarów mogą być na bieżąco wyświetlane na terminalu uruchomionym na PC. Do obsługi portu szeregowego użyto modułu bezpośredniego dostępu do pamięci. Oprogramowanie zostało zrealizowane w strukturze bez pętli zdarzeń. Wszystkie czynności związane z pomiarem odległości są wykonywane w przerwaniu timera.

Listing 1. Plik distsen.c

```
/*
 * NUCLEO F411/F401 with HC-SR04 distance sensor
 * gbm 12'2015
 */
#include "board.h"
#include "timing.h"

#define TRIG_BIT 10
#define ECHO_BIT 9
#define S_MFREQ  2  // measure freq. Hz
// us per timer clock
#define S_TCK_US  ((1000000 + 65536 * S_MFREQ - 1) / (65536 * S_MFREQ)
#define S_PERIOD  ((1000000 + S_TCK_US * S_MFREQ - 1) / (S_TCK_US * S_MFREQ))
#define TRIG_WIDTH (10 / S_TCK_US + 1)  //10 us
#define RST_START  (S_PERIOD - (30 / S_TCK_US + 1))  // 30 us
#define MAX_V_PING (1000 / S_TCK_US + 1)  // 1000 us

int main(void)
{
    // enable peripherals
    RCC->AHB1ENR = RCC_AHB1ENR_GPIOAEN | RCC_AHB1ENR_DMA1EN;
    RCC->APB1ENR = RCC_APB1ENR_USART2EN;
    RCC->APB2ENR = RCC_APB2ENR_TIM1EN;
    // port setup
    GPIOA->OTYPER = 1u << ECHO_BIT;        // Echo OD for reset
    GPIOA->AFR[0] = BF4(RX_BIT, 7) | BF4(TX_BIT, 7); // USART2
    GPIOA->AFR[1] = BF4(TRIG_BIT, 1) | BF4(ECHO_BIT, 1); // TIM1CH3,2
    GPIOA->MODER = GPIOA_MODER_SWD
        // TIM1CH3,2 - trig, echo
        | BF2(TRIG_BIT, GPIO_MODER_AF) | BF2(ECHO_BIT, GPIO_MODER_AF)
        // UART2 pins as AF
        | BF2(TX_BIT, GPIO_MODER_AF) | BF2(RX_BIT, GPIO_MODER_AF);
    // USART2 setup - PC connection
    USART2->BRR = (APB1_FREQ + BAUD_RATE / 2) / BAUD_RATE;
    USART2->CR3 = USART_CR3_DMAT;  // enable TX DMA
    USART2->CR1 = USART_CR1_TE | USART_CR1_UE;  // enable
    // TIM1 - distance sensor; CH1 - echo, CH2 - echo reset, CH3 - trig
    TIM1->PSC = APB2_TFREQ / 1000000 * S_TCK_US - 1;  // 10 us
    TIM1->ARR = S_PERIOD - 1;  // 2 Hz
    TIM1->CCR3 = TRIG_WIDTH;  // trigger pulse >= 10 us
    TIM1->CCR2 = RST_START;  // reset pulse - 20 us
    TIM1->CCMR1 = TIM_CCMR1_OC2M_PWM1 | TIM_CCMR1_CC1S_1;  // PWM mode 1, capture from TI2
    TIM1->CCMR2 = TIM_CCMR2_OC3M_PWM1;  // PWM mode 1
    TIM1->CCER = TIM_CCER_CC3E | TIM_CCER_CC2E | TIM_CCER_CC1NP | TIM_CCER_CC1P | TIM_CCER_CC1E;
// enable CH3&2 output, CH1 capture both edges
    TIM1->BDTR = TIM_BDTR_MOE;  // enable PWM outputs
    TIM1->DIER = TIM_DIER_CC1IE;  // enable capture 1 interrupt
    TIM1->CR1 = TIM_CR1_CEN;  // enable
    // interrupts and sleep
    NVIC_EnableIRQ(TIM1_CC_IRQn);
    SCB->SCR = SCB_SCR_SLEEPONEXIT_Msk;    // sleep while not in handler
    __WFI();
}

void TIM1_CC_IRQHandler(void)
{
    static char diststr[] = "    \r\n";
    static uint16_t       ping, echo;
    uint32_t ncap = TIM1->CCR1;
    if (TIM1->SR & TIM_SR_UIF)
    {
        TIM1->SR = ~TIM_SR_UIF;
        if (ping > TRIG_WIDTH && ping < MAX_V_PING && echo > ping)
        {
            uint32_t dist_mm = echo >= RST_START ? 9999
                : (echo - ping) * S_TCK_US * 10 / 58;
            // convert distance to string
            for (int i = 3; i >= 0; i --)
            {
                diststr[i] = dist_mm % 10 + '0';
                dist_mm /= 10;
            }  // start DMA xfer
            DMA1_Stream6->CR = 0;  // disable
            DMA1->HIFCR = DMA_HIFCR_CALLF6;  // clear stream flags
            DMA1_Stream6->PAR = (uint32_t)&USART2->DR;
            DMA1_Stream6->M0AR = (uint32_t)diststr;
            DMA1_Stream6->NDTR = sizeof(diststr) - 1;
            // USART2_TX, increment memory adress, mem->periph, enable
            DMA1_Stream6->CR = DMA_SxCR_CHSELV(4)
                | DMA_SxCR_MINC | DMA_SxCR_DIR_M2P | DMA_SxCR_EN;
        }
        ping = ncap;
    }
    else
        echo = ncap;
}
```

Projekt wykonano w środowisku Keil MDK-ARM v.5.17. Program główny pokazano na **listingu 1**, a cały folder projektu jest dostępny w pliku stm32_HC-SR04.zip.

Użycie timera

Do obsługi czujnika użyto timera TIM1. Okres timera określa okres wykonywania pomiarów. Wejście Trig czujnika jest sterowane z wyjścia CH3 pracującego w trybie PWM. Linia Echo jest połączona z linią timera CH2, która pracuje jako wyjście PWM (z otwartym drenem) i jednocześnie jako wejście dla kanału 1, działającego w trybie chwytania obu zboczy sygnału. Jedynym źródłem przerwań jest kanał 1 timera.

Podłączenie czujnika HC-SR04

Czujnik połączono z płytką Nucleo za pomocą 4 przewodów, zgodnie z **tabelą 2**.

Inicjowanie mikrokontrolera

Ponieważ w przykładowym programie nie mamy istotnego zapotrzebowania na moc obliczeniową, mikrokontroler STM32F411 pracuje z domyślnym źródłem taktowania – wewnętrznym generatorem RC o częstotliwości 16 MHz. Z taką też częstotliwością działają wszystkie jego peryferie. Częstotliwości taktowania poszczególnych bloków zostały zdefiniowane jako stałe w pliku nagłówkowym timing.h. Na wstępie sekwencji inicjującej następuje włączenie używanych w programie peryferiali – portu GPIOA, timera TIM1, interfejsu USART2 i sterownika bezpośredniego dostępu do pamięci DMA1. Programowanie portu GPIOA polega na ustawieniu trybu OD dla linii PA9, służącej jako wejście i wyjście sygnału Echo oraz na wybraniu odpowiednich funkcji AF dla linii interfejsu czujnika i USART2 (linie PA2, 3, 9, 10). Interfejs szeregowy USART2 zostaje zaprogramowany na szybkość 115200 b/s i transmisję z użyciem DMA.

Programowanie timera

Stałe czasowe potrzebne do zaprogramowania timera TIM1 zostały zdefiniowane na początku pliku distsen.c. Są to kolejno:

- S_MFREQ – częstotliwość, z którą są wykonywane kolejne pomiary odległości – zadawana przez programistę.
- S_TSC_US – okres zegara timera za preskalerem wyrażony w µs, wyliczony automatycznie, aby okres timera nie przekraczał 65536 cykli.
- S_PERIOD – okres timera wyrażony w cyklach, wyliczony automatycznie.
- TRIG_WIDTH – szerokość impulsu wyzwalającego pomiar – musi być ona nie mniejsza niż 10 µs.
- RST_START – czas od początku impulsu Trig do początku impulsu kasowania pomiaru na linii Echo, którego szerokość jest nie mniejsza od 30 µs; impuls kasowania kończy się wraz z końcem okresu timera.
- MAX_V_PING – maksymalny czas od początku impulsu Trig do początku impulsu Echo, przy którym pomiar może zostać uznany za ważny – służy on do odrzucenia nieregularnych odpowiedzi modułu HC-SR04 bezpośrednio po włączeniu zasilania układu; z doświadczeń wynika, że właściwą wartością jest tu 1000 µs.

Podczas programowania timera kolejno:

- Programujemy nastawy preskalera – rejestr PSC.
- Programujemy okres pomiarów – rejestr ARR.
- Programujemy czasy impulsów wyzwalania i kasowania pomiaru – rejestry CCR3 i CCR2.
- Ustawiamy tryb PWM dla kanału kasowania (CH2) i tryb *capture* z wejścia TI2 (będącego równocześnie wyjściem kanału CH2) dla kanału CH1 – rejestr CCMR1.
- Włączamy tryb PWM dla kanału wyzwalania pomiaru (CH3) – rejestr CCMR2.
- Włączamy wyjścia PWM dla kanałów CH3 i CH2 oraz programujemy tryb chwytania obu zboczy dla wejścia sygnału Echo – rejestr CCER.
- Włączamy globalnie wyjścia PWM – bit MOE w rejestrze BDTR.

Tabela 2. Dołączenie czujnika HC-SR04 do płytki Nucleo

Linia HC-SR04	Linia µC/ Nucleo	Złącze	Pozycja
Vcc	+5 V	CN7	18
Trig	PA10/TIM1_CH3	CN10	33
Echo	PA9/TIM1_CH2	CN10	21
GND	GND	CN7	20

- Włączamy przerwania od przechwycenia w kanale CH1.
- Włączamy timer.

Po zaprogramowaniu timera włączamy przerwanie timera, włączamy w procesorze tryb uśpienia przy wyjściu z obsługi wyjątku, a następnie usypiamy procesor.

Procedura obsługi przerwania timera

Procedura obsługi przerwania timera jest wywoływana dwukrotnie w każdym okresie pomiaru, przy wykryciu każdej zmiany stanu linii Echo. Pierwsza zmiana w okresie pomiarowym, z 0 na 1, następuje wraz z ustawieniem stanu aktywnego na linii Echo przez moduł HC-SR04. Druga zmiana – z 1 na 0 następuje w wyniku odebrania echa przez moduł HC-SR04 lub z powodu wygenerowania na linii Echo impulsu kasującego przez mikrokontroler.

Oprogramowanie nie bada stanu linii Echo. Przyjmujemy, że pierwsza zmiana stanu w cyklu pomiarowym jest zmianą z 0 na 1, a druga – z 1 na 0. Jako znacznik początku okresu jest używany znacznik przeładowania timera – bit UIF w rejestrze SR. Znacznik przerwania od zmiany stanu linii jest zerowany automatycznie z chwilą odczytu rejestru CCR1.

Programowa reakcja na wystąpienie pierwszej zmiany stanu linii Echo w okresie pomiarowym składa się z:

- Wyzerowania znacznika początku okresu pomiarowego.
- Weryfikacji poprawności pomiaru z poprzedniego okresu, wyliczenia odległości i rozpoczęcia transmisji danych przez USART.
- Zapamiętania czasu zmiany w zmiennej ping.

Przy drugiej zmianie stanu linii Echo w okresie pomiarowym czas zmiany jest zapamiętywany w zmiennej echo.

Bezpośrednio po włączeniu urządzenia, w jednym lub dwóch cyklach pomiarowych, mogą wystąpić anomalie w pracy czujnika. Poprawny cykl pomiarowy jest rozpoznawany po tym, że początek impulsu Echo występuje nie wcześniej niż po zakończeniu impulsu Trig i nie później niż 1 ms od początku cyklu pomiarowego (początku impulsu Trig), a druga zmiana stanu linii Echo jest rejestrowana z czasem późniejszym niż pierwsza (czyli pochodzi z obecnego, a nie z poprzedniego cyklu pomiarowego). Odległość czujnika od przeszkody jest obliczana wyłącznie po zweryfikowaniu poprawności danych.

Obliczenie odległości od przeszkody następuje na podstawie wartości zmiennych ping i echo. Ponieważ rozdzielczość czujnika jest lepsza od 1 cm, przyjęto, że jednostką odległości jest milimetr. Jeżeli zarejestrowany czas drugiej zmiany odpowiada początkowi impulsu kasującego, przyjmowana jest umowna odległość maksymalna (9999 mm), wykraczająca poza rzeczywisty zasięg czujnika. W przeciwnym razie odległość jest wyliczana na podstawie zarejestrowanej długości impulsu na linii Echo – różnicy wartości zmiennych echo i ping.

Po wyznaczeniu odległości w procedurze obsługi przerwania timera następuje konwersja odległości na 4-cyfrową postać tekstową. Następnie jest programowany sterownik DMA w celu przesłania łańcucha tekstowego zawierającego zmierzoną odległość przez moduł USART2, połączony z interfejsem VCOM ST-Link. Przesyłany tekst można obserwować na terminalu w komputerze PC.

Grzegorz Mazur