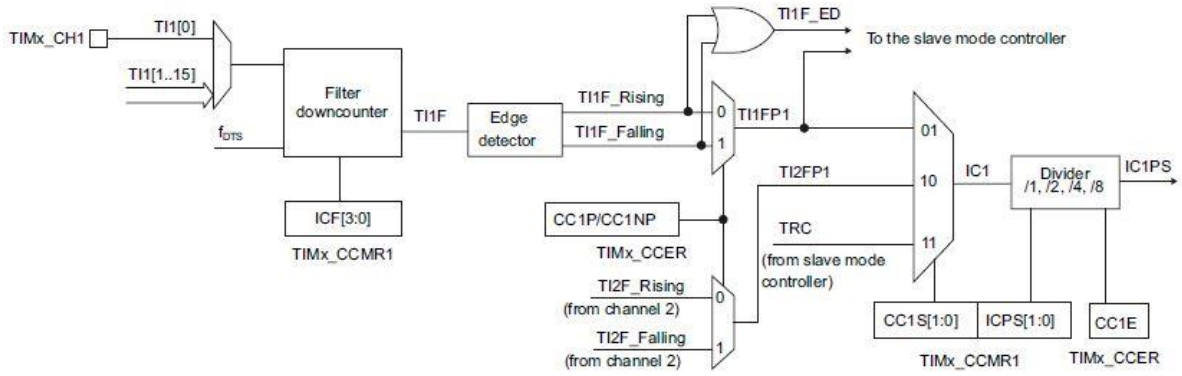
Zastosowanie liczników do pomiaru czasu, na przykładzie pomiaru odległości z wykorzystaniem czujnika HC-SR04

Wprowadzenie

Wejściami timera STM32 mogą być różne sygnały – zarówno fizyczne wejście na porcie GPIO, jak i wejścia z wewnętrznych układów peryferyjnych np. komparatorów. Sygnał z wejścia przekazywany jest do filtru. Mamy możliwość konfiguracji czasu trwania filtracji sygnału, dzięki czemu łatwo wyeliminujemy zakłócenia. Czas filtrowania ustawiamy w stosunku do sygnału podłączonego do timera. Na przykład, gdy przy przełączaniu sygnał wejściowy nie jest stabilny przez 5 wewnętrznych cykli zegara, musimy zaprogramować czas trwania filtra dłuższy niż te 5 cykli, czyli minimum 8. Przefiltrowany sygnał wchodzi do detektora zboczy. Timer może reagować na zbocza narastające, opadające lub na oba zbocza jednocześnie. Należy pamiętać, że timer generuje jedno wspólne zdarzenia dla każdego zbocza – na zbocze narastające i opadające flaga jest ta sama, co nie pozwala ich rozróżnić z poziomu timera. Sygnał na wyjściu detektora zboczy jest przekazywany do preskalera, gdzie mamy możliwość konfiguracji co które zbocze będzie powodowało zdarzenie. Sygnał z detektora może być również przekazany do innego timera ustawionego w trybie slave. Każde zdarzenie na wejściu timera po przejściu przez cały blok wchodzi do bloku głównego kanału Capture/Compare. Tam wystąpienie zdarzenia sprawia, że wartość z licznika (Counter) przekazywana jest do rejestru Capture/Compare, z którego może zostać odczytany i wykorzystany do dalszych obliczeń.



Rys. 1 Blok funkcjonalny timera / licznika

Budowa i zasada działania czujnika HC-SR04

Ultradźwiękowe czujniki odległości wykorzystują impulsy dźwięku ultradźwiękowego (dźwięk powyżej zakresu ludzkiego słuchu) do wykrywania odległości między nimi a pobliskimi obiektami stałymi. Czujniki składają się z dwóch głównych elementów: Nadajnik ultradźwiękowy – przesyła impulsy dźwięku ultradźwiękowego, działa z częstotliwością 40 kHz Odbiornik ultradźwiękowy – Odbiornik nasłuchuje przesyłanych impulsów. Jeśli je odbierze, wytwarza impuls wyjściowy, którego szerokość można wykorzystać do określenia odległości, którą przebył impuls.

HC-SR04 posiada następujące cztery złącza:

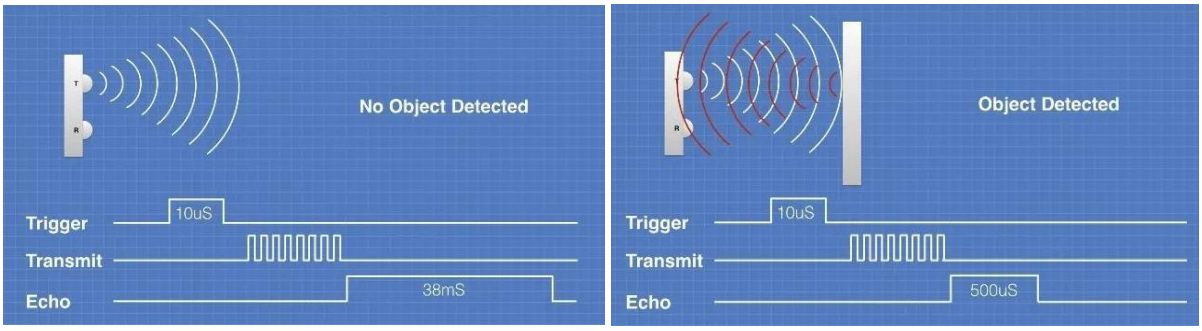
- VCC –zasilanie 5 lub 3,3V (w nowszych modelach),
- Trig – pin aktywujący wysłanie ciągu sygnałów ultradźwiękowych,
- Echo –pin wytwarzający impuls po odebraniu odbitego sygnału.
- GND – pin masy.

Długość impulsu jest proporcjonalna do czasu potrzebnego do wykrycia przesyłanego sygnału.



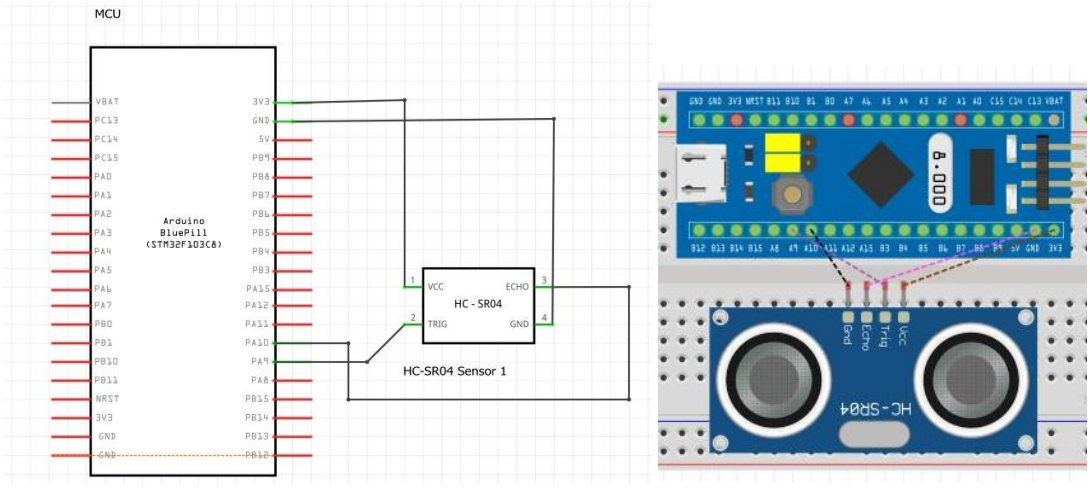
Rys.2 Wygląd typowego czujnika HC-SR04

Zasada działania czujnika ultradźwiękowego została przedstawiona poniższych rysunkach:



Rys. 3 Zasada działania czujnika ultradźwiękowego

Schemat ideowy podłączenia przetwornika do mikrokontrolera został przedstawiony poniżej:



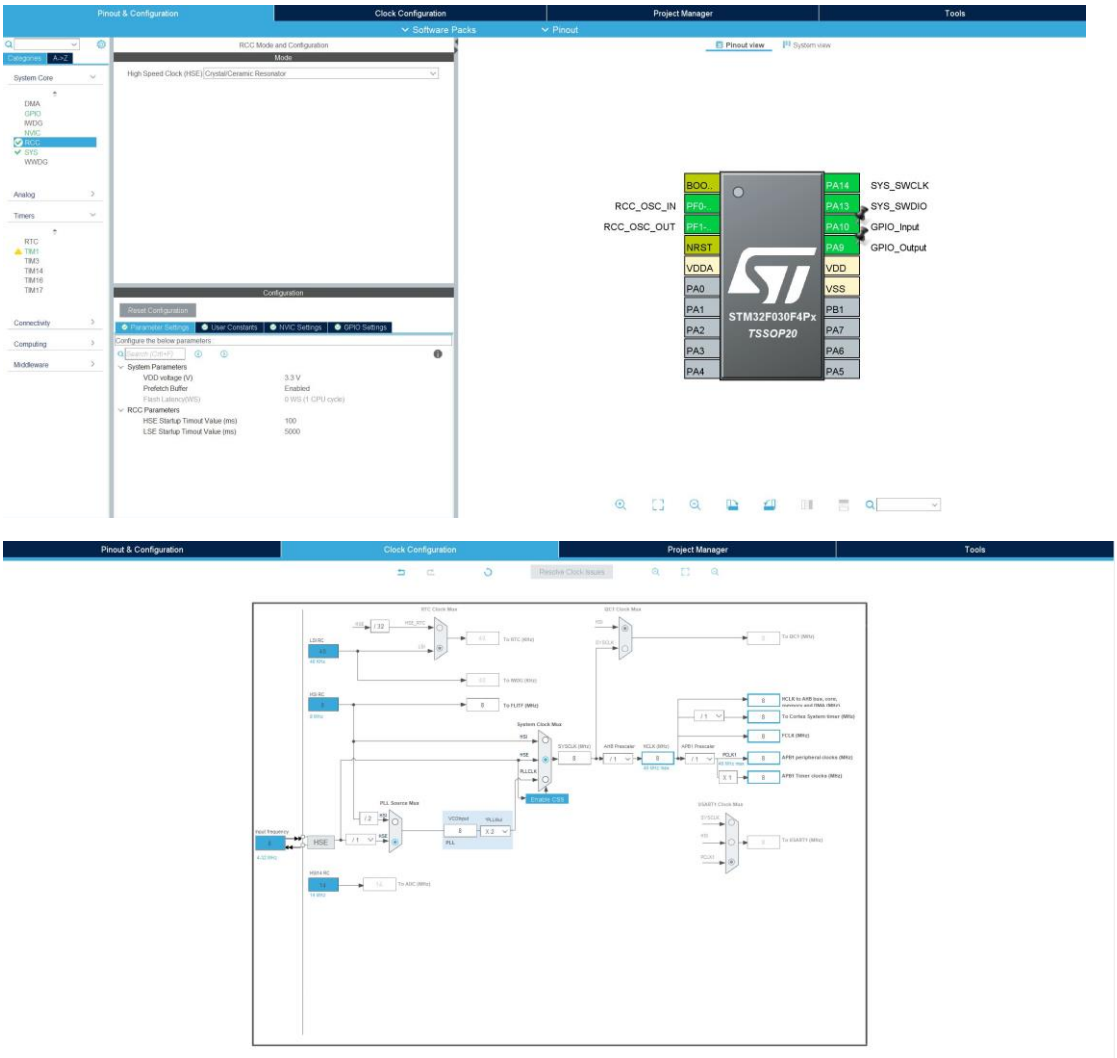
Rys. 4 Przykładowy schemat podłączenie czujnika do płytki prototypowej STM32

Zgodnie z dokumentacją techniczną czujnika HC-SR04, algorytm przetwarzania jest następujący:

- utrzymuj pin Trig w stanie wysokim przez co najmniej 10 us,
- moduł wyśle teraz 8-cyklową serię ultradźwięków o częstotliwości 40 kHz i wykryje, czy sygnał echo powrócił do odbiornika,
- jeśli sygnał powrócił, moduł wyśle impuls w stanie wysokim, którego długość będzie proporcjonalna do odległości od obiektu,
- odległość można obliczyć za pomocą następującego wzoru:
    - zasięg = czas wysokiego poziomu * prędkość rozchodzenia się dźwięku w powietrzu (340 m/s) / 2

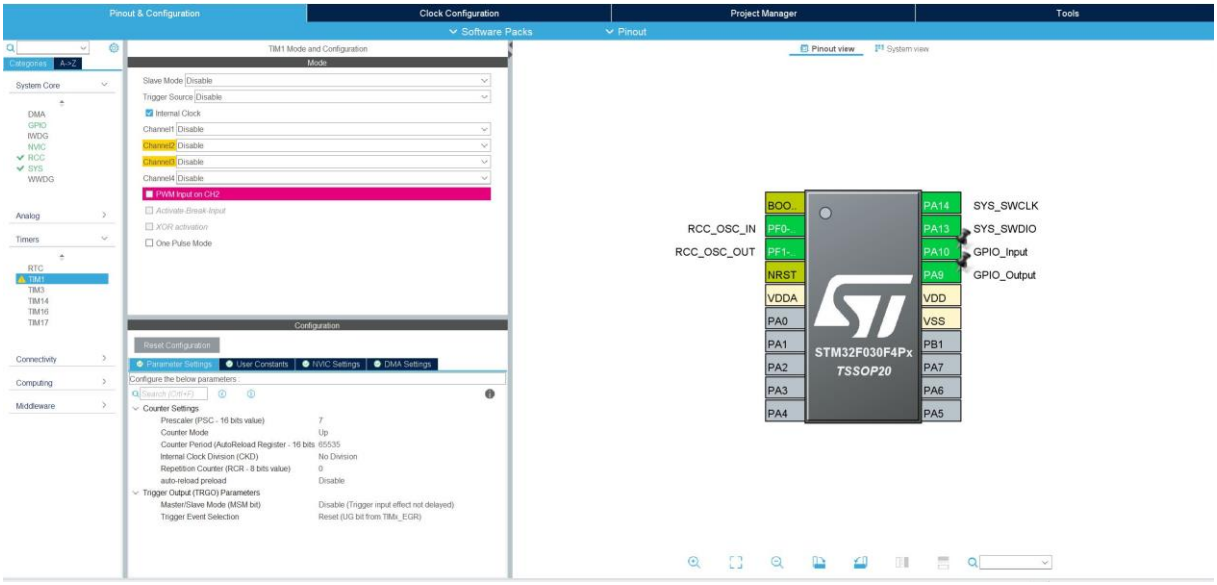
Przykładowy program do pomiaru odległości

W pierwszym kroku należy przełączy generator wewnętrzny RC na zewnętrzny generator kwarcowy, aby zwiększyć precyzje pomiarów czasu. Czy zegar generator kwarcowego będzie większy, to większą precyzję pomiaru możemy uzyskać.



Rys. 4 Przełączenie mikrokontrolera na taktowanie zegarem zewnętrznym

Ustawienia portów I/O do współpracy z czujnikiem HC-SR04



Rys 5. Ustawienia portów mikrokontrolera

Należy zwrócić szczególną uwagę na ustawienia preskalera, ustawiona wartość powinna być mniejsza o 1 od wartości zegara taktującego mikrokontroler. Dzięki temu będziemy mogli mierzyć czasy w zakresie mikrosekund, a takie przedziały czasowe są zwracane z czujnika ultradźwiękowego dla zakresu pomiarowego od ok. 2 cm do kilkudziesięciu centymetrów.

W utworzony domyślnym projekcie należy uzupełnić kod funkcji main o następujące fragmenty:

```
/* USER CODE BEGIN PV */
#define TRIG_PIN GPIO_PIN_9
#define TRIG_PORT GPIOA
#define ECHO_PIN GPIO_PIN_10
#define ECHO_PORT GPIOA
uint32_t pMillis;
uint32_t value1 = 0;
uint32_t value2 = 0;
uint16_t distance = 0;
/* USER CODE END PV */


/* USER CODE BEGIN 2 */
  HAL_TIM_Base_Start(&htim1);
  HAL_GPIO_WritePin(TRIG_PORT, TRIG_PIN, GPIO_PIN_RESET); // pull the TRIG pin low
  /* USER CODE END 2 */


while (1)
  {
    /* USER CODE END WHILE */
        HAL_GPIO_WritePin(TRIG_PORT, TRIG_PIN, GPIO_PIN_SET); // pull the TRIG pin
HIGH
        __HAL_TIM_SET_COUNTER(&htim1,0);
        while(__HAL_TIM_GET_COUNTER(&htim1)<10); // wait for 10 us
        HAL_GPIO_WritePin(TRIG_PORT, TRIG_PIN, GPIO_PIN_RESET); // pull the TRIG to
LOW
        pMillis = HAL_GetTick(); // used this to avoid infinite while loop
        // wait for echo pin to go HIGH
```

```
        while(!(HAL_GPIO_ReadPin(ECHO_PORT, ECHO_PIN)) && pMillis + 10 >
HAL_GetTick());
        value1 = __HAL_TIM_GET_COUNTER(&htim1);
        pMillis = HAL_GetTick(); // used this to avoid infinite while loop
        // wait for echo pin to go LOW
        while((HAL_GPIO_ReadPin(ECHO_PORT, ECHO_PIN)) && pMillis + 50 >
HAL_GetTick());
        value2 = __HAL_TIM_GET_COUNTER(&htim1);
        distance = (value2 - value1) * 0.034/2;
        HAL_Delay(50);
    /* USER CODE BEGIN 3 */
  }
```

Następnie należy uruchomić program w trybie debugera, wartość odległości będzie można odczytać ze zmiennej „distance”.