NewPing

Knihovna umožňuje současnou obsluhu většího množství ultrazvukových senzorů. Rozsáhlou technickou podporu naleznete na wiki jejího autora: https://bitbucket.org/teckel12/arduino-new-ping/wiki/Home

Třída NewPing

Konstruktor

*Konstruktor vytvoří jednu nebo více instancí třídy NewPing, aktivuje piny pro připojení senzorů a nastaví způsob jejich řízení. V následujících příkladech je použita instance pojmenovaná **sonar1**.*

Syntaxe:

```
NewPing sonar1([trigger_pin], [echo_pin], [max_cm_distance]);
```

Parametry:

trigger_pin (*byte*) – [*nepovinné, výchozí hodnota je 12*] číslo pinu, který spouští měření.
echo_pin (*byte*) – [*nepovinné, výchozí hodnota je 11*] číslo pinu pro příjem měřícího impulsu.
max_cm_distance (*unsigned int*) – [*nepovinné, výchozí hodnota je 500 cm*] maximální měřená vzdálenost.

Návratová hodnota:

Funkce nic nevrací.

Příklad:

```
NewPing sonar1(12, 11, 200);
```

Vytvoří instanci třídy NewPing, pojmenovanou sonar1. Spouštěcí signál bude generován na pinu 12, délka vstupního pulsu se měří na pinu 11. Největší očekávaná vzdálenost překážky je nastavena na 200 cm (výchozí hodnota je 500 cm).

Poznámka:

Pokud je použit ultrazvukový senzor v jednovodičovém zapojení (např. SRF06), jsou parametry trigger_pin a echo_pin shodné.

Členské funkce (metody)

ping()

Vyšle ultrazvukový puls a vrací čas, uplynulý mezi jeho odesláním a příjmem echa v mikrosekundách (µs) nebo číslo 0 (nula), pokud se měření nezdařilo.

Syntaxe:

```
sonar1.ping();
```

Parametry:

Funkce nemá žádné parametry.

Návratová hodnota:

(*unsigned int*)

Čas uplynulý mezi odesláním měřícího pulsu a příjmem echa v mikrosekundách (µs).

Příklad:

```
// proměnná, do které se bude ukládat návratová hodnota funkce
int echoTime;

echoTime = sonar1.ping();
```

ping_cm()

Vyšle ultrazvukový puls a vrací vzdálenost překážky v centimetrech (cm) nebo číslo 0 (nula), pokud se měření nezdařilo.

Syntaxe:

```
sonar1.ping_cm();
```

Parametry:

Funkce nemá žádné parametry.

Návratová hodnota:

(*unsigned int*)

Vzdálenost od překážky v centimetrech.

Příklad:

```
// proměnná, do které se bude ukládat návratová hodnota funkce
int distance_cm;

distance_cm = sonar1.ping_cm();
```

ping_in()

Vyšle ultrazvukový puls a vrací vzdálenost překážky v palcích (inch) nebo číslo 0 (nula), pokud se měření nezdařilo.

Syntaxe:

```
sonar1.ping_in();
```

Parametry:

Funkce nemá žádné parametry.

Návratová hodnota:

(*unsigned int*)

Vzdálenost od překážky v palcích.

Příklad:

```
// proměnná, do které se bude ukládat návratová hodnota funkce
int distance_in;

distance_in = sonar1.ping_in();
```

ping_median()

Odešle parametrem iterations zvolený počet ultrazvukový pulsů a vrátí střední hodnotu všech správných měření v mikrosekundách (µs). Výsledky měření, které jsou mimo rozsah nastavený při vytváření instance ignoruje.

Syntaxe:

```
sonar1.ping_median([iterations]);
```

Parametr:

iterations (*byte*) – [*nepovinné, výchozí hodnota je 5*] počet měření

Návratová hodnota:

Funkce nic nevrací.

convert_cm()

Převádí výsledek měření v mikrosekundách na vzdálenost v centimetrech.

Syntaxe:

```
sonar1.convert_cm(echoTime);
```

Parametr:

echoTime – (*unsigned int*) čas mezi odesláním měřícího pulsu a příjmem echa v mikrosekundách (µs)

Návratová hodnota:

(*unsigned int*)
Vzdálenost překážky v centimetrech.

convert_in()

Převádí výsledek měření v mikrosekundách (µs) na vzdálenost v palcích (inch).

Syntaxe:

```
sonar1.convert_in(echoTime);
```

Parametr:

echoTime – (*unsigned int*) čas mezi odesláním měřícího pulsu a příjmem echa v mikrosekundách (µs)

Návratová hodnota:

Funkce nic nevrací.

ping_timer()

Vyšle ultrazvukový puls a zavolá uživatelem definovanou funkci, která výsledek měření zpracuje.

Syntaxe:

```
sonar1.ping_timer(function);
```

Parametr:

function – jméno volané (uživatelem definované) funkce

Návratová hodnota:

Funkce nic nevrací.

check_timer()

Zkontrolujte, zda se echo vrátilo v rámci limitu vzdálenosti, nastaveného při vytváření instance parametrem max_cm_distance.

Syntaxe:

```
sonar1.check_timer();
```

Parametry:

Funkce nemá žádné parametry.

Návratová hodnota:

TRUE – pokud se echo vrátilo v nastaveném limitu

FALSE – v opačném případě

Příklad:

```
// proměnná, do které se bude ukládat návratová hodnota funkce
bool jePrekazka;

jePrekazka = sonar1.check_timer();
```

timer_us()

Opakované volání parametrem určené funkce v mikrosekundách.

Syntaxe:

```
sonar1.timer_us(frequency, function);
```

Parametry:

frequency – (*unsigned int*) frekvence opakovaného volání určené funkce

function – jméno volané (uživatelem definované) funkce

Návratová hodnota:

Funkce nic nevrací.

timer_ms()

Opakované volání parametrem určené funkce.

Syntaxe:

```
sonar1.timer_ms(frequency, function);
```

Parametry:

frequency – (*unsigned long*) frekvence opakovaného volání určené funkce v milisekundách (ms).
function – jméno volané funkce.

Návratová hodnota:

Funkce nic nevrací.

timer_stop()

Zastavuje časovač.

Syntaxe:

```
sonar1.timer_stop();
```

Parametry:

Funkce nemá žádné parametry.

Návratová hodnota:

Funkce nic nevrací.

Příklady

NewPing

```
#include <NewPing.h>

const int TRIGGER_PIN = 12;
const int ECHO_PIN = 11;
const int MAX_DISTANCE = 200;

NewPing sonar1(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

void setup()
{
  Serial.begin(115200);
}

void loop()
{
  delay(50);
  Serial.print("Vzdálenost:");
  Serial.print(sonar1.ping_cm());
  Serial.println(" cm");
}
```

Simple NewPing Sketch

Dvacet měření za sekundu.

```
#include <NewPing.h>

const int TRIGGER_PIN = 12;
// Arduino pin tied to trigger pin on the ultrasonic sensor.
const int ECHO_PIN = 11;
// Arduino pin tied to echo pin on the ultrasonic sensor.
const int MAX_DISTANCE = 200;
// Maximum distance we want to ping for (in centimeters).
// Maximum sensor distance is rated at 400-500cm.

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
// NewPing setup of pins and maximum distance.

void setup()
{
  Serial.begin(115200);
```

```
  // Open serial monitor at 115200 baud to see ping results.
}

void loop()
{
  delay(50);
  // Wait 50 ms between pings (about 20 pings/sec).
  // 29 ms should be the shortest delay between pings.
  Serial.print("Ping: ");
  Serial.print(sonar.ping_cm());
  // Send ping, get distance in cm and print result (0 = outside set distance range)
  Serial.println("cm");
}
```

Ping 3 Sensors Sketch

Obsluha tří dálkoměrů 20x za sekundu.

```
#include <NewPing.h>

const int SONAR_NUM = 3;
// Number of sensors.

const int MAX_DISTANCE 200
// Maximum distance (in cm) to ping.

// Sensor object array.
NewPing sonar[SONAR_NUM] =
{
  // Each sensor's trigger pin, echo pin, and max distance to ping.
  NewPing(4, 5, MAX_DISTANCE),
  NewPing(6, 7, MAX_DISTANCE),
  NewPing(8, 9, MAX_DISTANCE)
};

void setup()
{
  // Open serial monitor at 115200 baud to see ping results.
  Serial.begin(115200);
}

void loop()
{
  // Loop through each sensor and display results.
  for (uint8_t i = 0; i < SONAR_NUM; i++)
  {
    // Wait 50 ms between pings (about 20 pings/sec).
```

```
    // 29 ms should be the shortest delay between pings.
    delay(50);
    Serial.print(i);
    Serial.print("=");
    Serial.print(sonar[i].ping_cm());
    Serial.print("cm ");
  }
  Serial.println();
}
```

Event Timer Sketch

Tento příklad předvádí použití funkce NewPing `ping_timer`*, která využívá přerušení Timer2 k získání doby trvání měřícího pulsu. Výhodou použití této funkce oproti standardní funkci* `ping` *je, že umožňuje řídit váš program událostmi, a tedy zdánlivě provádět dvě činnosti naráz. Je třeba mít stále na zřeteli, že funkce* `ping_timer` *používá Timer2, takže ovlivňuje všechny další funkce nebo knihovny, které Timer2 také používají. Například funkce PWM na pinech 3 a 11 v Arduino Uno (piny 9 a 11 v Arduino Mega) a knihovna Tone.*

```
#include <NewPing.h>

const int TRIGGER_PIN = 12;
// Arduino pin tied to trigger pin on ping sensor.
const int ECHO_PIN = 11;
// Arduino pin tied to echo pin on ping sensor.
const int MAX_DISTANCE = 200;
// Maximum distance we want to ping for (in centimeters).
// Maximum sensor distance is rated at 400-500cm.

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
// NewPing setup of pins and maximum distance.

unsigned int pingSpeed = 50;
// How frequently are we going to send out a ping (in milliseconds).
// 50ms would be 20 times a second.

unsigned long pingTimer;
// Holds the next ping time.

void setup()
{
  Serial.begin(115200);
  // Open serial monitor at 115200 baud to see ping results.

  pingTimer = millis();
  // Start now.
```

```
}

void loop()
{
  // Notice how there's no delays in this sketch to allow you to do other
  // processing in-line while doing distance pings.
  if (millis() >= pingTimer)
  {
    // pingSpeed milliseconds since last ping, do another ping.
    pingTimer += pingSpeed;
    // Set the next ping time.
    sonar.ping_timer(echoCheck);
    // Send out the ping, calls "echoCheck" function every 24us
    // where you can check the ping status.
  }
  // Do other stuff here, really. Think of it as multi-tasking.
}

void echoCheck()
{
  // Timer2 interrupt calls this function every 24 us
  // where you can check the ping status.
  // Don't do anything here!
  if (sonar.check_timer())
  {
    // This is how you check to see if the ping was received.
    // Here's where you can add code.
    Serial.print("Ping: ");
    Serial.print(sonar.ping_result / US_ROUNDTRIP_CM);
    // Ping returned, uS result in ping_result, convert to cm with US_ROUNDTRIP_CM.
    Serial.println("cm");
  }
  // Don't do anything here!
}
```

Timer Median Sketch

Výpočet střední hodnoty všech přijatých měřících impulsů funkcí ping_timer()

```
#include <NewPing.h>

const int ITERATIONS = 5;
// Number of iterations.
const int TRIGGER_PIN = 12;
// Arduino pin tied to trigger pin on ping sensor.
const int ECHO_PIN = 11;
// Arduino pin tied to echo pin on ping sensor.
```

```
const int MAX_DISTANCE 200
// Maximum distance (in cm) to ping.
const int PING_INTERVAL = 33;
// Milliseconds between sensor pings
// (29ms is about the min to avoid cross-sensor echo).

unsigned long pingTimer[ITERATIONS];
// Holds the times when the next ping should happen for each iteration.

unsigned int cm[ITERATIONS];
// Where the ping distances are stored.

uint8_t currentIteration = 0;
// Keeps track of iteration step.

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
// NewPing setup of pins and maximum distance.

void setup()
{
  Serial.begin(115200);
  pingTimer[0] = millis() + 75;
  // First ping starts at 75ms, gives time for the Arduino to chill before starting.
  for (uint8_t i = 1; i < ITERATIONS; i++)
    // Set the starting time for each iteration.
    pingTimer[i] = pingTimer[i - 1] + PING_INTERVAL;
}

void loop()
{
  for (uint8_t i = 0; i < ITERATIONS; i++)
  {
    // Loop through all the iterations.
    if (millis() >= pingTimer[i])
    {
      // Is it this iteration's time to ping?
      pingTimer[i] += PING_INTERVAL * ITERATIONS;
      // Set next time this sensor will be pinged.
      if (i == 0 && currentIteration == ITERATIONS - 1) oneSensorCycle();
      // Sensor ping cycle complete, do something with the results.
      sonar.timer_stop();
      // Make sure previous timer is canceled before starting
      // a new ping (insurance).
      currentIteration = i;
      // Sensor being accessed.
      cm[currentIteration] = 0;
      // Make distance zero in case there's no ping echo for this iteration.
      sonar.ping_timer(echoCheck);
```

```
      // Do the ping (processing continues, interrupt will call
      // echoCheck to look for echo).
    }
  }
  // Other code that *DOESN'T* analyze ping results can go here.
}

void echoCheck()
{
  // If ping received, set the sensor distance to array.
  if (sonar.check_timer())
    cm[currentIteration] = sonar.ping_result / US_ROUNDTRIP_CM;
}

void oneSensorCycle()
{
  // All iterations complete, calculate the median.
  unsigned int uS[ITERATIONS];
  uint8_t j, it = ITERATIONS;
  uS[0] = NO_ECHO;
  for (uint8_t i = 0; i < it; i++)
  {
    // Loop through iteration results.
    if (cm[i] != NO_ECHO)
    {
      // Ping in range, include as part of median.
      if (i > 0)
      {
        // Don't start sort till second ping.
        for (j = i; j > 0 && uS[j - 1] < cm[i]; j--)
          // Insertion sort loop.
          uS[j] = uS[j - 1];
        // Shift ping array to correct position for sort insertion.
      }
      else j = 0;
      // First ping is sort starting point.
      uS[j] = cm[i];
      // Add last ping to array in sorted position.
    }
    else it--;
    // Ping out of range, skip and don't include as part of median.
  }
  Serial.print(uS[it >> 1]);
  Serial.println("cm");
}
```

15 Sensors Example Sketch

*Tento kód byl autory knihovny úspěšně použit ke komunikaci s 15 ultrazvukovými senzory. Počet senzorů v projektu je možno upravit změnou konstanty SONAR_NUM a počtu objektů NewPing v poli "sonar". Rovněž je třeba změnit vývody pro jednotlivé senzory pro objekty NewPing. Senzory jsou spouštěny v intervalu 33 ms. Jeden cyklus obsluhy všech senzorů tedy trvá 495 ms (33 * 15 = 495 ms). Výsledky jsou odesílány do funkce "oneSensorCycle", která v tomto příkladu pouze zobrazuje údaje o vzdálenosti. Váš projekt by normálně zpracovával výsledky senzorů v této funkci (například by rozhodl, zda robot potřebuje zatočit, a zavolal by funkci zatočit). Mějte na paměti, že tento příklad je řízen událostmi. Váš kompletní program proto musí být napsán tak, aby neobsahoval žádné příkazy "delay" a aby doba běhu cyklu loop() byka kratší než 33 ms. Pokud jiné procesy trvají déle než 33 ms, budete muset zvýšit PING_INTERVAL.*

```
#include <NewPing.h>

const int SONAR_NUM = 15;  // Number or sensors.
const int MAX_DISTANCE = 200;  // Max distance in cm.
const int PING_INTERVAL = 33;  // Milliseconds between pings.

unsigned long pingTimer[SONAR_NUM];  // When each pings.
unsigned int cm[SONAR_NUM];  // Store ping distances.
uint8_t currentSensor = 0;  // Which sensor is active.

// Sensor object array.
NewPing sonar[SONAR_NUM] =
{
  NewPing(41, 42, MAX_DISTANCE),
  NewPing(43, 44, MAX_DISTANCE),
  NewPing(45, 20, MAX_DISTANCE),
  NewPing(21, 22, MAX_DISTANCE),
  NewPing(23, 24, MAX_DISTANCE),
  NewPing(25, 26, MAX_DISTANCE),
  NewPing(27, 28, MAX_DISTANCE),
  NewPing(29, 30, MAX_DISTANCE),
  NewPing(31, 32, MAX_DISTANCE),
  NewPing(34, 33, MAX_DISTANCE),
  NewPing(35, 36, MAX_DISTANCE),
  NewPing(37, 38, MAX_DISTANCE),
  NewPing(39, 40, MAX_DISTANCE),
  NewPing(50, 51, MAX_DISTANCE),
  NewPing(52, 53, MAX_DISTANCE)
};

void setup()
{
```

```
  Serial.begin(115200);
  pingTimer[0] = millis() + 75;  // First ping start in ms.
  for (uint8_t i = 1; i < SONAR_NUM; i++)
    pingTimer[i] = pingTimer[i - 1] + PING_INTERVAL;
}

void loop()
{
  for (uint8_t i = 0; i < SONAR_NUM; i++)
  {
    if (millis() >= pingTimer[i])
    {
      pingTimer[i] += PING_INTERVAL * SONAR_NUM;
      if (i == 0 && currentSensor == SONAR_NUM - 1)
        oneSensorCycle();  // Do something with results.
      sonar[currentSensor].timer_stop();
      currentSensor = i;
      cm[currentSensor] = 0;
      sonar[currentSensor].ping_timer(echoCheck);
    }
  }
  // The rest of your code would go here.
}

// If ping echo, set distance to array.
void echoCheck()
{
  if (sonar[currentSensor].check_timer())
    cm[currentSensor] = sonar[currentSensor].ping_result / US_ROUNDTRIP_CM;
}

// Do something with the results.
void oneSensorCycle()
{
  for (uint8_t i = 0; i < SONAR_NUM; i++)
  {
    Serial.print(i);
    Serial.print("=");
    Serial.print(cm[i]);
    Serial.print("cm ");
  }
  Serial.println();
}
```

Příklad zapojení