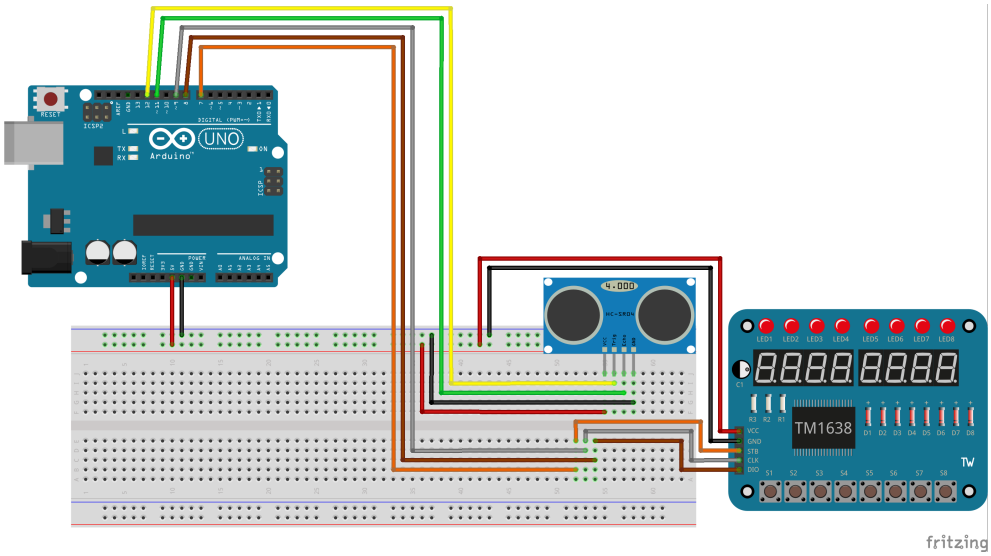
Ultradźwiękowy miernik odległości

Krzysztof Pakaszewski
Piotr Seemann
Wiktor Mendalka
Numer zespołu: 36
Informatyka II rok EAIiIB 2018/2019

Czerwiec 2019

1 Schemat



2 Opis algorytmu

Lista kroków:

1. Zacznij algorytm.
2. Wyświetl migający napis w postaci '————' (osiem kresek poziomych).
3. Po naciśnięciu przycisku S1 wygaś wyświetlacz.

4. Dokonaj 50 pomiarów czasu biegu fali dźwiękowej do przeszkody i z powrotem, następnie podziel wynik przez 2, żeby otrzymać czas biegu fali do przeszkody i oblicz odległość od przeszkody mnożąc wynik przez prędkość dźwięku ($distance = \frac{duration \cdot 0.034}{2}$).
5. Oblicz średni dystans.
6. Wyświetl końcowy wynik pomiaru w formacie xxx.x
7. Po naciśnięciu przycisku S2 skasuj wynik i powróć do punktu 2.
8. Zakończ algorytm.

Algorytm działa poprawnie dla zakresu 30-200 cm.

3 Opis programu

Zmienne:

1. digits - jest to tablica, która po odowołaniu się do niej za pomocą cyfry i, zwraca binarny zapis na wymagane dla wyświetlenia tej cyfry LEDy
2. dist - przechowuje obliczony dystans
3. strobe, clock, data, trigPin, echoPin - wartości stałe przechowujące kody pinów, w które wpięte są przewody
4. showValue, test - zmienne pomocnicze pozwalające zachowywać stan wyświetlania, tj. zmieniają wartość, gdy wciśnięte zostaną przeciski mające zmienić tryb pracy

Metody:

1. sendCommand - wysyła podaną wartość na strobe, wartość to kod operacji do wykonania
2. reset - ustawia wartość flag dla ledów na 0, czyli sprawia, że nic się nie wyświetla
3. setup - ustawia poszczególne piny tak, by miały możliwość komunikacji danych
4. readButtons - zwraca, w postaci liczby binarnej, status przycisków (czy są wciśnięte)
5. setLed - ustawia status diody LED o danym kodzie
6. measure - wykonuje pojedyńczy pomiar
7. distance - wykonuje serię pomiarów przy użyciu measure
8. showDistance - wyświetla na ekranie zadaną liczbę
9. defaultScreen - wyświetla ekran domyślny, tj. 8 pionowych kresek
10. loop - główna funkcja programu, uruchamiająca się co cykl

4 Biblioteki

Brak zewnętrznych bibliotek.

5 Kod źródłowy

```
#include <math.h>

const int strobe = 7;
const int clock = 9;
const int data = 8;
const int trigPin = 12;
const int echoPin = 11;

//.GFE DCBA
//0110 1101
int digits[]= {0x3f,0x06,0x5b,0x4f,0x66,0x6d,
    0x7d,0x07,0x7f,0x6f};
int dist;
bool test =true;
bool showValue= false;
void sendCommand(uint8_t value)
{
  digitalWrite(strobe, LOW);
  shiftOut(data, clock, LSBFIRST, value);
  digitalWrite(strobe, HIGH);
}

void reset()
{
  sendCommand(0x40); // set auto increment mode
  digitalWrite(strobe, LOW);
  shiftOut(data, clock, LSBFIRST, 0xc0);   // set starting address to 0
  for(uint8_t i = 0; i < 16; i++)
  {
    shiftOut(data, clock, LSBFIRST, 0x00);
  }
  digitalWrite(strobe, HIGH);
}

void setup()
{
  pinMode(strobe, OUTPUT);
  pinMode(clock, OUTPUT);
  pinMode(data, OUTPUT);
```

```
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  Serial.begin(9600); // Starts the serial communication

  sendCommand(0x8f);  // activate
  reset();
}

uint8_t readButtons(void)
{
  uint8_t buttons = 0;
  digitalWrite(strobe, LOW);
  shiftOut(data, clock, LSBFIRST, 0x42);

  pinMode(data, INPUT);

  for (uint8_t i = 0; i < 4; i++)
  {
    uint8_t v = shiftIn(data, clock, LSBFIRST) << i;
    buttons |= v;
  }

  pinMode(data, OUTPUT);
  digitalWrite(strobe, HIGH);
  return buttons;
}

void setLed(uint8_t value, uint8_t position)
{
  pinMode(data, OUTPUT);

  sendCommand(0x44);
  digitalWrite(strobe, LOW);
  shiftOut(data, clock, LSBFIRST, 0xC0 + (position << 1));
  shiftOut(data, clock, LSBFIRST, value);
  digitalWrite(strobe, HIGH);
}

// 0100 0000
double measure(){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
```

```
  // Reads the echoPin, returns the sound wave travel time in microseconds
  long duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  double distance= duration*0.034/2;
  return distance;
  }
double distance(){
  double sum =0;
  for(int a =0; a<50; a++){
      sum+=measure();
  }
    return sum/50;
}


void showDistance(int dist){
    int l = floor(log10(dist));

    for(uint8_t position = 8-l; position < 8; position++)
    {
      int wyswietl = dist/pow(10,8-position);
      dist = dist % (int)(pow(10,8-position));

      uint8_t maska =digits[wyswietl];

      if (position == 6)
      {
        maska = maska | 0x80;
      }

      setLed(maska, position);
      }
}

void defaultScreen(){
  for(uint8_t position = 0; position < 8; position++)
    {
      setLed(0x40, position);
     }
  }

void loop()
{
 //reset();
 uint8_t buttons = readButtons();
```

```
 uint8_t button1 = buttons & 0x01;
 uint8_t button2 = buttons & 0x02;


 if(button1 && test){
    reset();
    showValue = true;
    dist = (int)(distance()*100);
    test= false;
  }
  if(!button1){
    test= true;
    }

   if(button2){
      showValue= false;
    }
   if(showValue){
     showDistance(dist);
   }
   else{
         defaultScreen();
         delayMicroseconds(200);
         reset();
   }
}
```