

Senzorové systémy

Určení polohy pomocí ultrazvuku

Ondřej Jirman

Zadání

Vymyslet metodu plynulého měření polohy pomocí ultrazvuku za použití šuffíkových součástí.

1 Teorie

Polohu neznámého bodu (X) v protoru můžeme určit pokud známe vzdálenost tohoto bodu od tří známých bodů (A, B, C) neležících na jedné přímce vyřešíme tuto soustavu rovnic:

$$|AX| = a \quad (1)$$

$$|BX| = b \quad (2)$$

$$|CX| = c \quad (3)$$

Zjednodušíme si práci a řekněme, že bod A má polohu (0, 0, 0), bod B má polohu (0, 1, 0), bod C má polohu (1, 0, 0) a bod X má polohu (x, y, z).

Rovnice tak získají následující tvar (kartézská soustava souřadnic):

$$a^2 = x^2 + y^2 + z^2 \quad (4)$$

$$b^2 = x^2 + (y - 1)^2 + z^2 \quad (5)$$

$$c^2 = (x - 1)^2 + y^2 + z^2 \quad (6)$$

Tyto rovnice budou mít za daných předpokladů 2 řešení:

$$x = -1/2 c^2 + 1/2 + 1/2 a^2 \quad (7)$$

$$y = -1/2 b^2 + 1/2 + 1/2 a^2 \quad (8)$$

$$z = 1/2 \sqrt{-2 - c^4 + 2 c^2 + 2 c^2 a^2 + 2 b^2 a^2 - 2 a^4 - b^4 + 2 b^2} \quad (9)$$

a

$$x = -1/2 c^2 + 1/2 + 1/2 a^2 \quad (10)$$

$$y = -1/2 b^2 + 1/2 + 1/2 a^2 \quad (11)$$

$$z = -1/2 \sqrt{-2 - c^4 + 2 c^2 + 2 c^2 a^2 + 2 b^2 a^2 - 2 a^4 - b^4 + 2 b^2} \quad (12)$$

Tato řešení se liší pouze v hodnotě souřadnice z bodu X, což je logické, protože body A, B a C leží v rovině z=0.

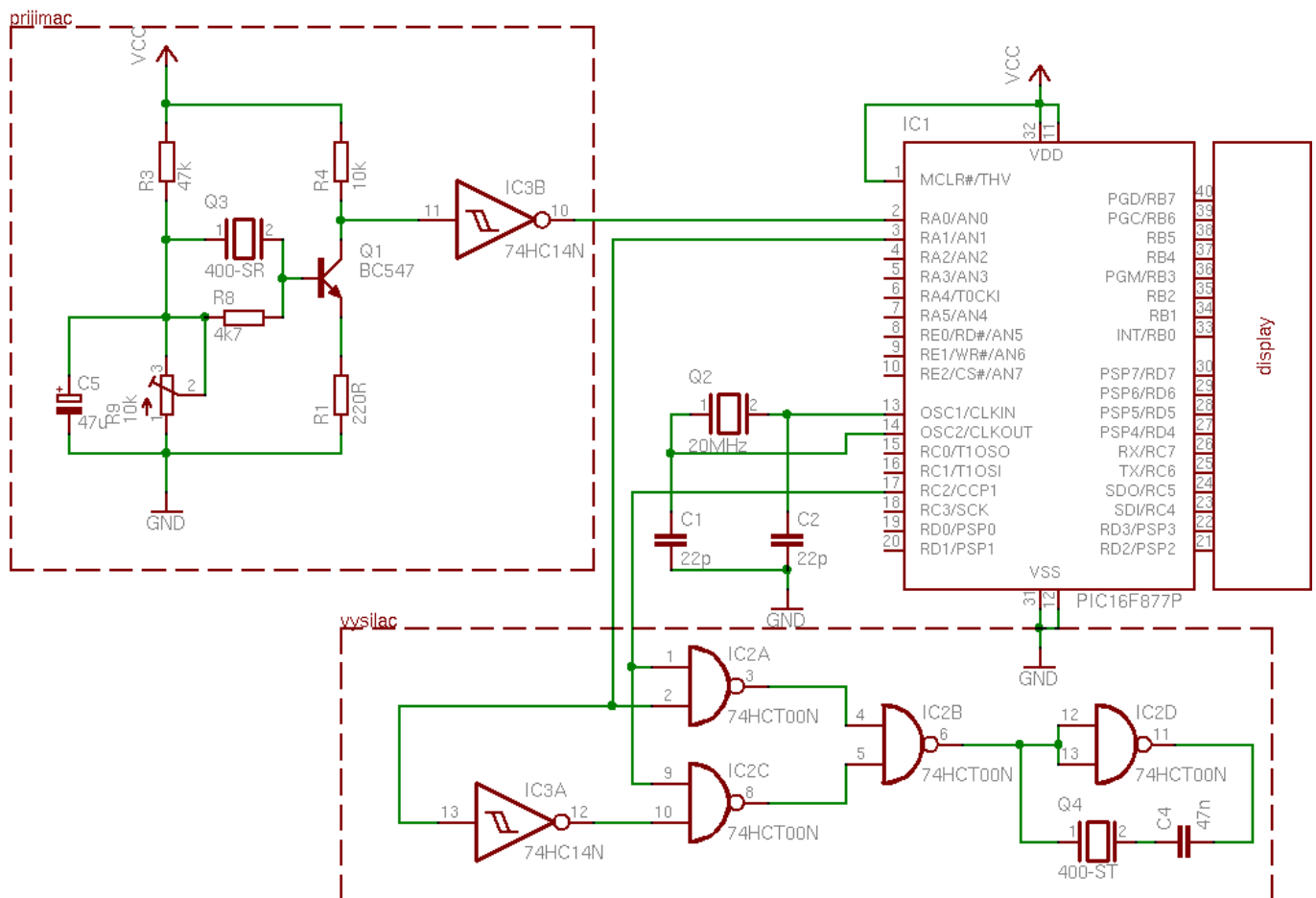
Nyní již ke zjištění kartézských souřadnic bodu X v prostoru spočátkem v bodu A, stačí změřit vzdálenosti a, b a c.

Měření

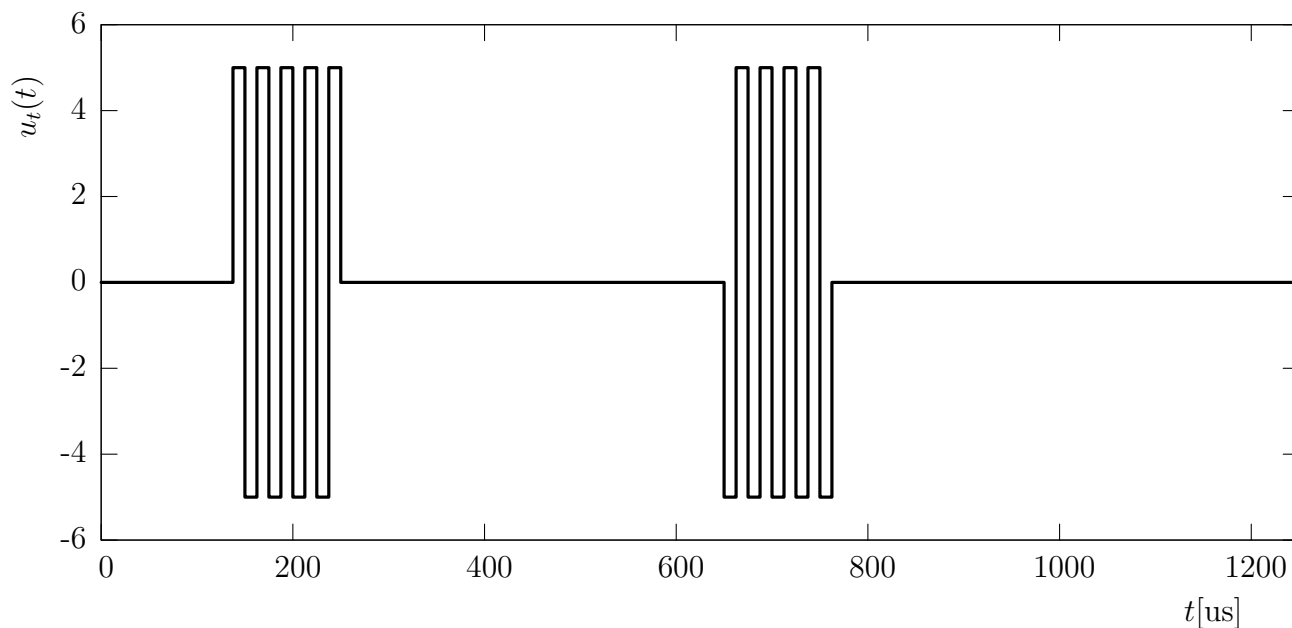
Známe přibližnou rychlost zvuku ve vzduchu, která je 340 m/s. Vzdálenosti tedy můžeme měřit jako doby od vyslání signálu z bodu X do přijetí signálu v bodech A,B a C (Ta, Tb a Tc).
 Vztah mezi vzdáleností a časem je:

$$l = vt \tag{13}$$

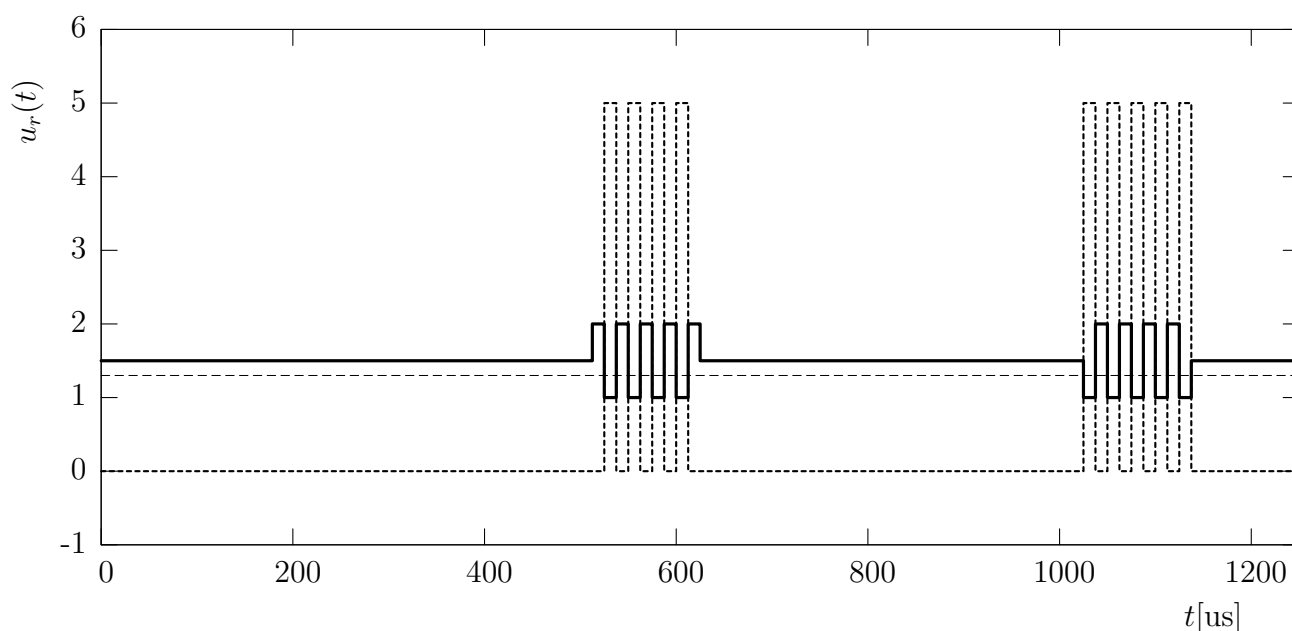
Navržené měřicí zapojení je na následujícím obrázku. Na obr. 2 je časový průběh vysílaného impulsu a na obr. 3 je znázorněna funkce přijímacího obvodu. Vysílací část obvodu se skládá z invertoru fáze signálu a multiplexeru, pomocí kterého se požadovaná fáze vybírá. Přijímací část obvodu se skládá z zesilovače a komparátoru, který funguje jako detektor impulsů. Činnost vysílacího obvodu je vidět na obr. 1. Činnost přijímacího obvodu na obr. 2.



Obr. 1: Vysílání



Obr. 2: Příjem



Závěr

Z obr. 3 je vidět hlavní problém navrženého zapojení. Tím problémem je, že přijímací obvod detekuje pouze zápornou půlvalu přijímaného impulsu. Takže pokud k přijímači dorazí nejprve kladná půlvalna, tak bude ignorována. Toto způsobuje velmi nepříjemnou, systematickou chybu měření, která závisí na vzdálenosti vysílače od přijímače. Lidsky řečeno, při pomalém oddalování vysílače od přijímače se měřená vzdálenost nemění plynule, ale nepřijemně kolísá v rozmezí délky vlny (zhruba 9mm). Tato chyba je v zapojení odstraněna pomocí střídavé změny fáze vysílaného impulsu o π a průměrování 10 úspěšných měření zpoždění impulsu.

Dále je nutné odstranit zpoždění (asi 25 μs), které vzniká kvůli tomu, že vysílač i přijímač mají jistou

setrvačnost. Pokud budu předpokládat, že obě dvě součástky mají stejné zpoždění, pak to vychází zhruba na půl délky vysílané vlny, což neodporuje zdravému rozumu.

Příloha: Měřicí program

```
#include "samples.h"

static u16 deltas[3];

static u32 lengths[3];

static const u8 delta_map[] = {
    0, 0, 1, 0, 2, 0, 1, 0
};
static const u8 mask_map[] = {
    ~0x01, ~0x02, ~0x04
};

static void wait_13ms()
{
    TMRO = 0;
    TOIF = 0;
    while (!TOIF);
}

// not really :)
static void wait_200ms()
{
    u8 t = 30/13;
    while (t--)
        wait_13ms();
}

static bit pulse_phase = 0;

static void start_pulse()
{
    TMR2ON = 1; // turn on tmr2
}

static void stop_pulse()
{
    if (pulse_phase)
    {
        RA1 = 0;
        pulse_phase = 0;
    }
}
```

```
else
{
    RA1 = 1;
    pulse_phase = 1;
}
TMR2ON = 0; // turn off tmr2
}

static u8 measure_deltas()
{
    u8 mask = 0x01; // detect bit 0 H level
    TMR0 = 0; // reset tmr0
    TOIF = 0;
    start_pulse();
    TMR1L = 0;
    TMR1H = 0;

    while (!TOIF)
    {
        u8 pa = PORTA & mask;
        if (pa)
        {
            u16 t0 = TMR1L | (TMR1H<<8);
            do
            {
                u8 b = delta_map[pa];
                deltas[b] = t0;
                mask &= mask_map[b];
                if (mask == 0)
                    goto got_all_mics;
                pa &= mask;
            } while (pa);
        }
    }

    got_all_mics:
    stop_pulse();
    wait_13ms(); // wait till the pulse travels away
    return mask == 0;
}

#define DELAY 830
#define LCONST 20
#define AVGLEN 10

static u8 measure_lengths()
{
```

```
u8 loss = 0;
u8 good = 0;

lengths[0] = 0;
while (loss < AVGLEN && good < AVGLEN)
{
    if (measure_deltas())
    {
        lengths[0] += deltas[0];
        good++;
    }
    else
        loss++;
}
// make avg and convert to metric units
if (loss != AVGLEN)
{
    deltas[0] = (lengths[0]/AVGLEN - DELAY) / LCONST;
}
if (deltas[0] > 500)
    deltas[0] = 0;
return loss != AVGLEN;
}

char str[12];

void main()
{
    // initialize display
    glInit();
    glDALInit();
    glFSetFont(&font_arialn_16);

    // port a setup
    ADCON1 = 0x06; // set porta pins to digital io mode
    TRISA = 0xfd; // set porta to input mode
    // setup pulse generator
    CCP1CON = 0x0c; // PWM mode
    PR2 = 124; // period = (PR2 + 1) * 4 * Tosc * TMR2 prescale
    CCPR1L = 62; // duty cycle = CCPR1L * 4 * Tosc * TMR2 prescale
    TRISC2 = 0; // output direction on ccp1 pin (rc2)
    T2CON = 0; // no pre/post scaler
    // setup timeout timer
    OPTION = 0xc7; // setup tmr0 (max prescaler, timeout 13ms)
    // setup delay measurement timer
    T1CON = 0x01;
```

```
while (1)
{
    if (measure_lengths())
    { // all mics detected impulse (or noise, at least)
        // clear display
        glClear();
        sprintf(str, "%c: %u mm", 'x', deltas[0]);
        // draw string with measured value
        glFDrawString(1, 20, str);
        glLine(0, 45, deltas[0]/4, 45);
        glLine(0, 46, deltas[0]/4, 46);
        glLine(0, 47, deltas[0]/4, 47);
        glDALFlush();
    }
    wait_200ms();
}
}
```