

The contest entry F2037

Artificial Vision System for the Blind

The various systems have been developed to help blind individuals orienting in the surround space. The range of these systems is very wide and stretches from the simplest, inexpensive devices to the complex, multi-component vision systems which transforming the optical camera images into electric signals and transmit them to the cerebral cortex via implanted electrodes. However, these systems are very complex, expensive and are seldom used.

The dedicated ultrasound range finders are more popular. The existing range finders measure the distance to the object using ultrasound beam and represent it via variable tone sound signals. This approach has limitations because the individual must remember the lookup table between sound tone and the distance. In addition, these range finders are incapable of detecting moving objects such as cars or people which reduce safety.

This contest entry describes the vision system which helps the blind orientating in the surround space. As contrasted to existing systems, the proposed device combines the usefulness, simplicity to use together with low cost. Device measures the distance to the object and “visualizes” the measurement results as special slider position which can be easily touched by blind people. Moreover, the device estimates the neighbored objects speed using Doppler Effect and generates the proportional to the objects speed number of sound signals. As result, the blind individual with the proposed system can travel efficiently and safely. The system technical specifications are listed in the table below:

Table 1. The system technical specifications

The human interface controls	Power switch, Sound on/off switch, Multifunctional buzzer, Stepper motor driven slider
The power supply	2 AA size batteries, 2.7-3.6 V
Current consumption (3.2V supply)	40-50 ma motor off, Near 300 ma when motor on
Distance measuring range	20 cm – 3 m
Distance measuring accuracy	8% with no calibration
Speed measuring range	0.25 m/s – 8 m/s
Measurement cycle duration	0.3 s

The Flowchart and Schematic

The device flowchart is illustrated by Figure 1. The system consist of the ultrasound transmitter TX and receiver RX, the CPU-controlled carrier generator to form the ultrasound carrier signal, the transmitter driver to amplify the carrier signal to the piezoelectric transmitter acceptable levels, the signal mixer to select the Doppler frequency shift during speed measurements, synchronous rectifier to rectify the incoming signal during distance measurements, multiplexer, low-pass filter to suppress the high-frequency mixer/rectifier products, analog-to-digital converter (ADC), CPU for system control, an buzzer and the slider stepper motor driver with motor.

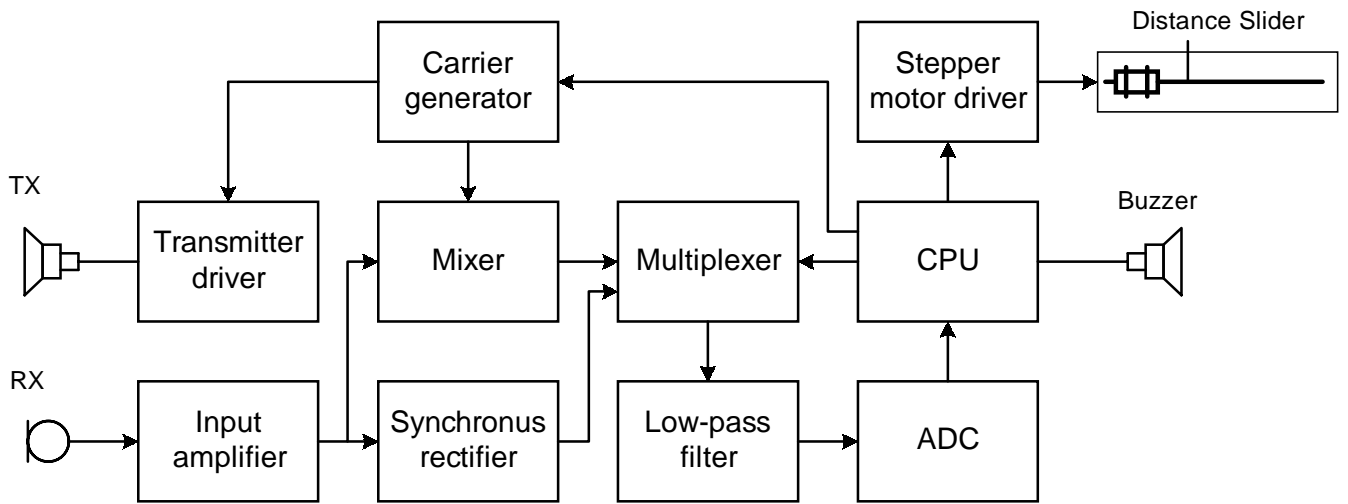


Figure 1. The vision system flowchart

The system schematics are illustrated by Figure 2 and Figure 3.

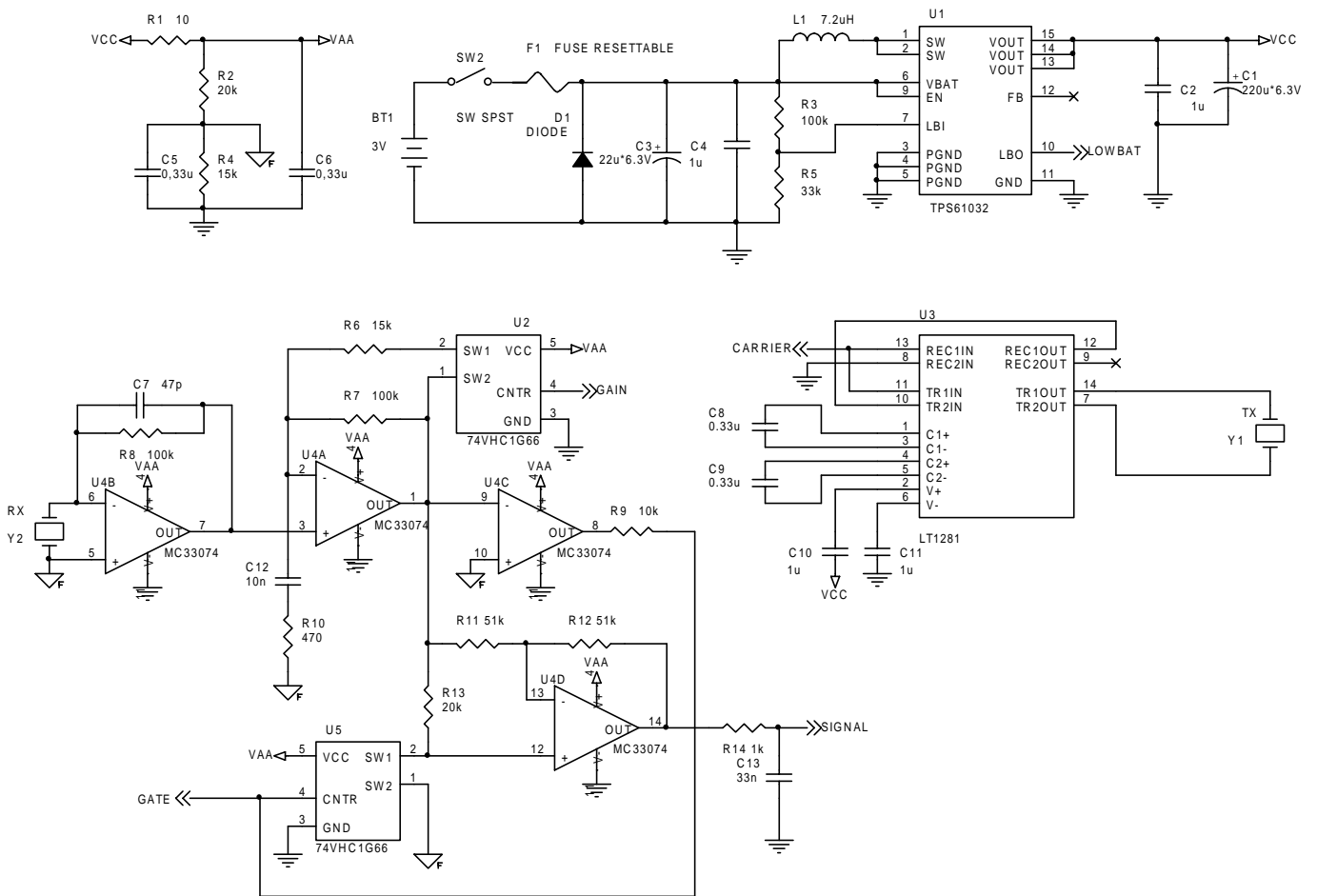


Figure 2. The vision system analog part schematic

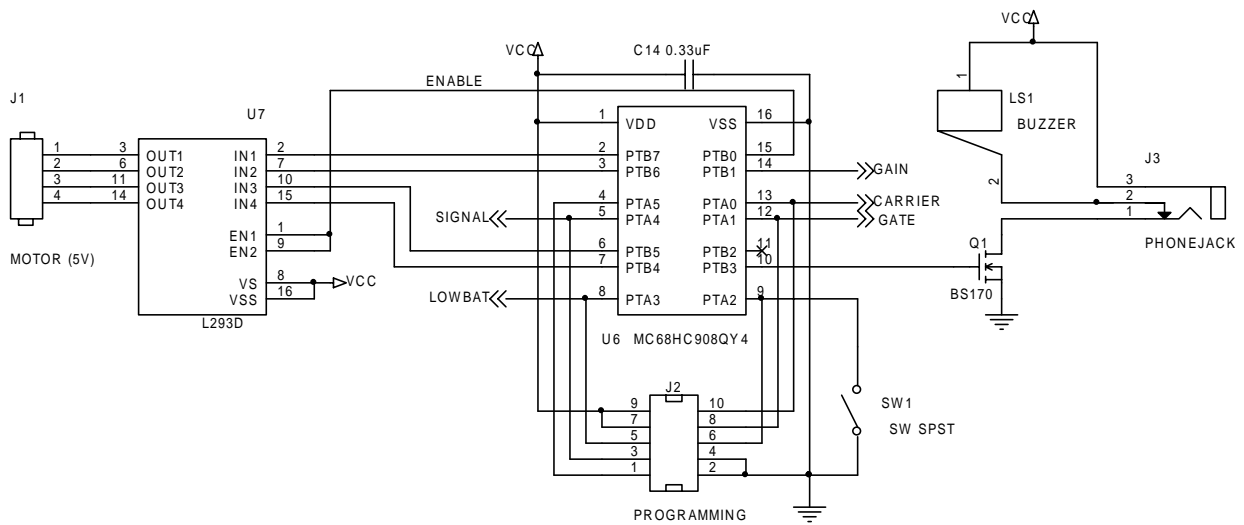


Figure 3. The vision system CPU part schematic

The device photographs

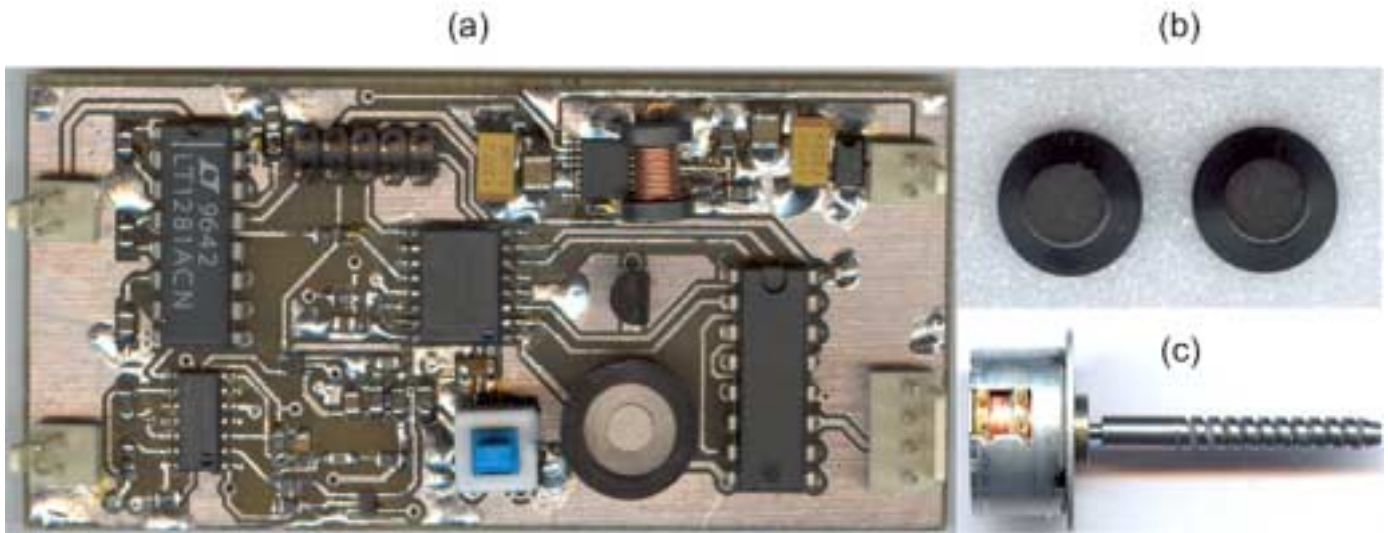


Figure 4. The system PCB (a), ultrasound sensors (b) and stepper motor with worm screw (c)

A sample code: The code below is used to measure distance to object (Measure_Distance function) and filter the results using the median filter (Measure_Distance_Filtered function).

```

word Measure_Distance(void)
{
    int var = MAX_TIME, max_time;
    byte code, max = 0;
    static byte gain = GAIN_LOW;

    ADICLK = 0x40;
    ADSCR = 0x22;

    Generator_Gate(RECTIFIER_MODE);
    GAIN_CONTROL_BIT = gain;

    delay(DELAY_DEAD);

```

```

Generator_Start();
delay(DELAY_PULSE);
Generator_Stop();
delay(DELAY_DEAD);

while(var)
{
    code = ADR;
    if (code > max)
    {
        max = code;
        max_time = var;
    }

    var--;
    while (!ADSCR_COCO);
}

max_time = MAX_TIME - max_time - INIT_DELAY;

var = 0;
code = BACKGROUND_AVERAGES;
while(code--)
{
    var += ADR;
    while (!ADSCR_COCO);
}
var >>= BACKGROUND_SHIFT;

if ((max - MAX_DELTA) > var) gain = GAIN_LOW;
if ((max - MAX_THLD) < var) gain = GAIN_HIGH;

if (((max - MAX_THLD) < var) || (max_time < 0) )
{
    return MEASURE_DISTANCE_ERROR;
}

return max_time;
}
word Measure_Distance_Filtered(void)
{
    word buffer[MEDIAN_FILTER_SIZE], temp;
    byte i, changed;

    for (i=0; i<MEDIAN_FILTER_SIZE; i++) buffer[i] = Measure_Distance();

    do {
        changed = 0;
        for (i = 0; i < MEDIAN_FILTER_SIZE-1; i++)
            if (buffer[i] > buffer[i+1])
            {
                changed = 1;

                temp = buffer[i];
                buffer[i] = buffer[i+1];
                buffer[i+1] = temp;
            }
    } while (changed);

    return buffer[MEDIAN_FILTER_SIZE>>1];
}

```