Vehicle Reversing Assistant

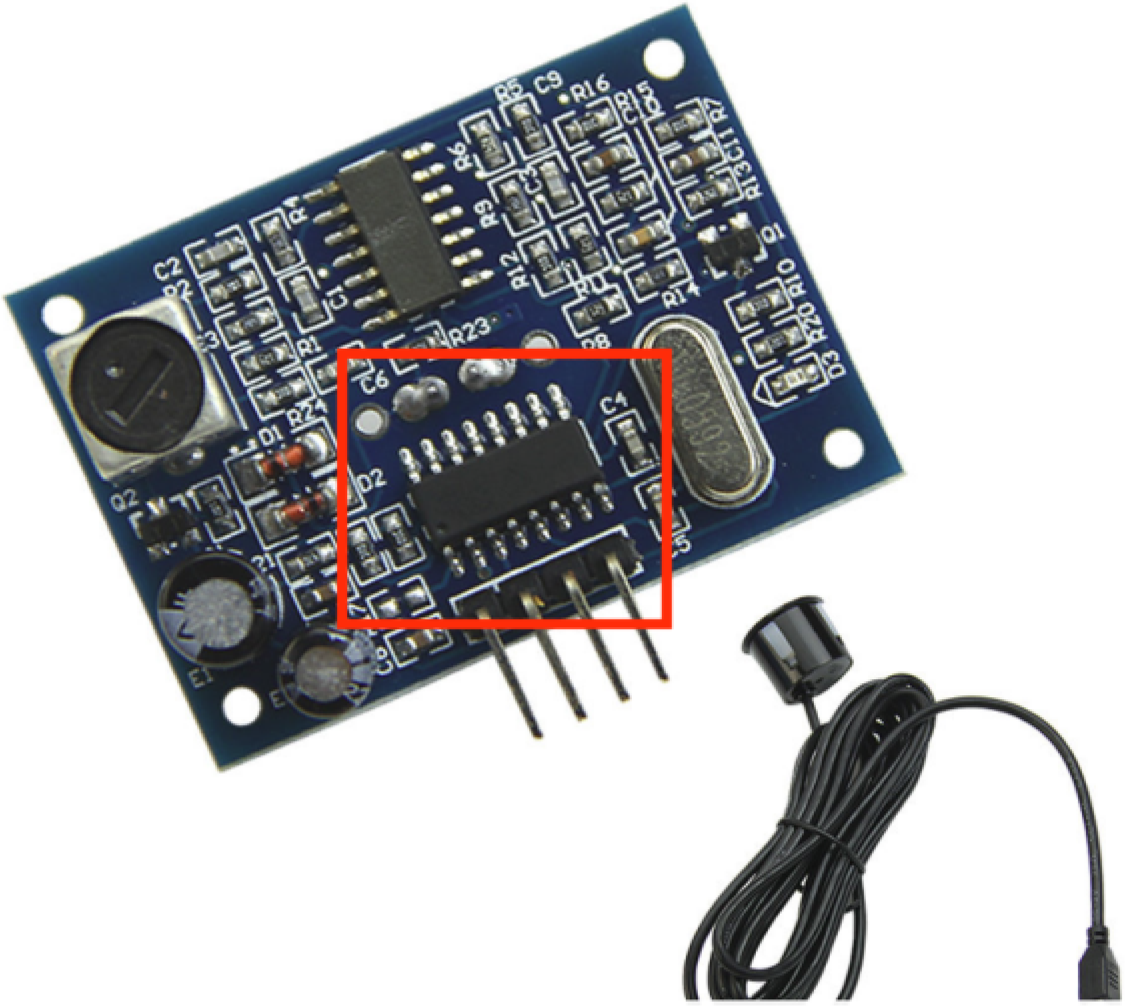
The following is a tiny trinket to retro-fit your vehicle with reversing sensors (or fix a crappy Chinese kit) using waterproof 40KHz ultra-sound sensor modules available off eBay etc. for about £5.

Be careful with buying these modules - there are two versions that I can find. One has a trigger and output pin so you can get a reading when you want it, the other type has a serial output and is free-running. The code below works with the former. I have found it very difficult to identify which is which before hand.

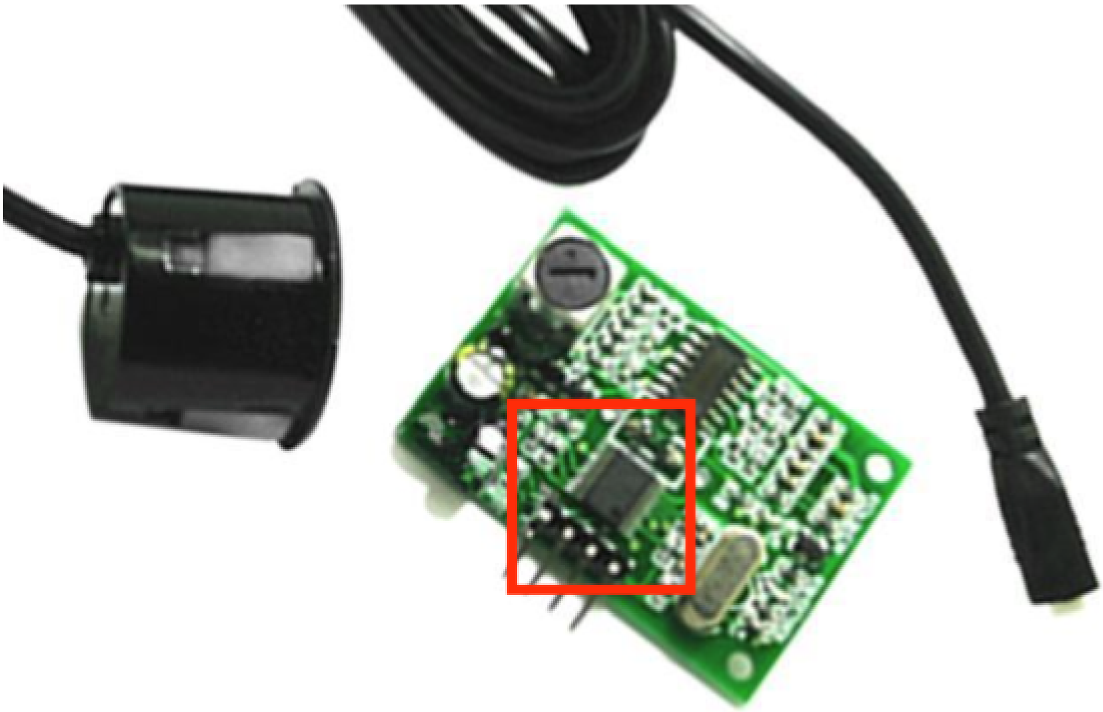
The free-running type presents a problem because despite claims of anti-jamming, all these modules suffer from picking up ultrasound pulses from neighbouring modules and will provide false readings. To illustrate, consider that you are some distance from an object and a pulse is initiated. A neighbouring module also sends a pulse shortly after but it “hears” the return from the first module - it can't tell this is not the reflection of its own pulse and now calculates a much shorter distance than reality. OK this is fail safe but it makes for a lot of false readings and generally results in a useless system - you have to be able to control when the pulses are sent and wait for any return. If the modules are free-running, it's chaos. If you want/have to use free-running ones, you are going to have to switch their power with a FET and give them time to die between readings or find the reset pin of the microcontroller and hold them in reset when not in use (and give them time to wake up after) - Do-able but a pain.

There is a way to tell these modules apart, but you need them in front of you; if you apply power and the LED flickers constantly, it is free-running. These are fine for solitary use and give a serial output with the direct reading of the distance. If the LED gives a single flash, it is waiting to be triggered and these are the ones you can use in groups. Here are pictures of the two types - note there is a difference in the built-in microcontroller. You will have to source the appropriate modules. The pictures below might help you to find the right ones first time.

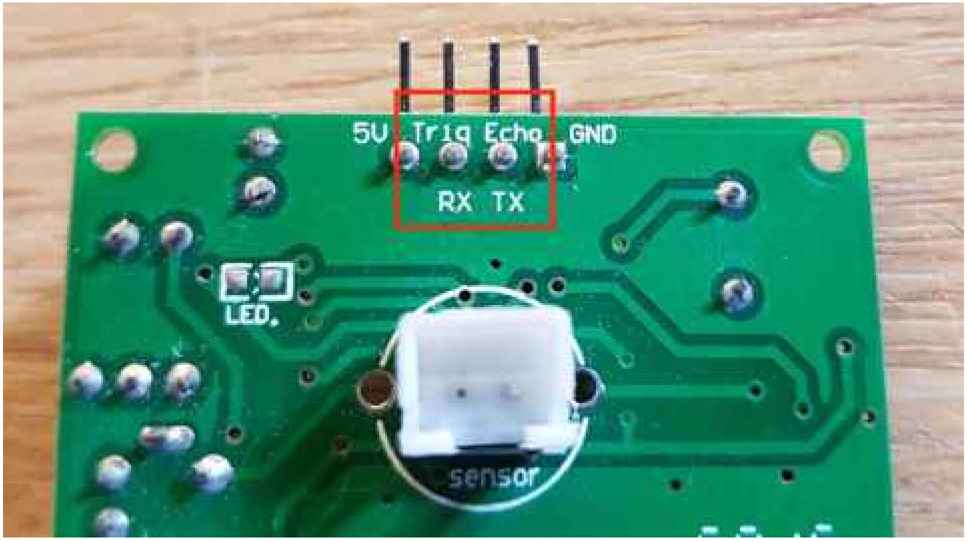
Triggered:



Free-running:



Confusingly the silk screen marks the pins on both types as Rx/Tx and Trig/Echo so no help there :o(



My particular situation has three sensors. The trigger and outputs of each sensor need to be connected to appropriate pins of your MicroMite.

The code works by measuring all three sensors and then choosing the shortest distance. This is then compared to the timer and if the timer is greater then a beep is issued and the timer set to zero. The closer the object, the faster the beeps. There is a lock-up zone of 400mm - any closer than this and the sounder emits a continuous tone. This has the nice effect that the value for the timer changes in real time and any changes are reflected by the rapidity of the beeps fairly quickly. Also, because the time between beeps is the same number of milliseconds as millimeters from the nearest obstacle, you become accustomed to judging distance - 1000mm means the beeps are a second apart. I have become quite good at judging distance with this.

You need a 12V to 3.3V PSU and if you connect this to the supply to the reversing lights of your vehicle, the circuit starts beeping as you need it.

Beeps are emitted via PWM1A but you can change this if you need. You will need to provide a suitable audio device - a piezo sounder with a FET driver at 12V is good.

Dependencies

Distance() CFunction - This is in the embedded modules section of the MMBasic distribution.

```
    CPU 48 'need to be fairly quick
    Option Base 0
    Option Baudrate 9600 'debug output is available to the console
    Option AutoRun On

    Const Ver=1.3
    Const LockDist=500'safe distance in mm
    Const Beeper=18

    'pins for trigger and return on the three distance sensors
    Dim Integer n,m,SP(5)=(14,15,16,17,21,22),SR(2)'SP is the pins for the
Distance function for 28pin MM
    Dim Float x

    'Pin 18 is the beeper @5v OC output
    Setpin Beeper,DOUT,OC 'we are going to sink current from the beeper
~10mA
    'some beeps at start
    For n=1 to 2
        Pin(Beeper)=0:Pause 50:Pin(Beeper)=1:Pause 50
    Next

    Pause 200

    Do
        WatchDog 2000
        Print "Sensor readings..."
        For n=0 To 5 Step 2'three pairs of pins
            x=Distance(SP(n),SP(n+1))

            Select Case x
                Case (0-2)
                    m=-2'sensor failure (-2) - beep out the sensor number as
long beep, then up to 3 short beeps
                    Pin(Beeper)=0:Pause 500:Pin(Beeper)=1
                    pause 500
                    for m=1 To (n/2)+1
                        Pin(Beeper)=0:Pause 250:Pin(Beeper)=1
                        Pause 250
                    Next
                Case (0-1)
```

```
                    m=3000 'no obstruction so default to 3m
                Case Else
                    m=10*Distance(SP(n),SP(n+1)) 'mm
            End Select
            Print x;m,
            SR(n\2)=m
        Next
        n=Min(SR(0),SR(1),SR(2))'choose closest reading will beep constantly
if a sensor failure (-2)
        Print "min=";n;"mm"


        If n<LockDist Then'< safe distance then lockup
            if n>0 then Pin(Beeper)=0'constant tone but not if an error
exists
        ElseIf Timer>=Max(n-LockDist,0) Then'otherwise beep
            Pin(Beeper)=0:Pause 250:Pin(Beeper)=1'fixed length beeps
            Timer=0
        EndIf
        Pause 90
    Loop

    Print "Broken out of main thread, Forcing watchdog restart."
    Do:loop

    CFunction DISTANCE(integer, integer) float
        0000002D 27BDFFF8 AFBF0004 00852023 03E42021 ACC40000 8FBF0004
03E00008
        27BD0008 27BDFFE0 AFBF001C 00002021 3C059D00 24A50040 27A60010
0411FFF1
        00000000 8FA30010 3C029D00 8C4200BC 3C049D00 24840310 0040F809
00832021
        8FBF001C 03E00008 27BD0020 000410C0 00041A00 00621823 00031180
00431823
        00641821 00031980 3C029D00 8C420000 3C047735 34849400 8C420000
0082001B
        004001F4 00002012 0064001B 008001F4 03E00008 00001012
        27BDFFD8 AFBF0024 AFB40020 AFB3001C AFB20018 AFB10014 AFB00010
00809021
        3C029D00 8C420000 8C430000 3C020098 34429680 0062102B 10400003
00A08821
        0411FFCA 00000000 8E220000 14400003 3C109D00 8E420000 AE220000
8E020010
        8E240000 24050002 0040F809 2406000E 8E02001C 8E440000 0040F809
24050005
        8E020010 8E440000 24050008 0040F809 00003021 8E02001C 8E440000
0040F809
        24050006 8E020004 0040F809 24040014 8E02001C 8E440000 0040F809
24050005
        8E020004 0040F809 24040032 8E020010 8E240000 24050002 0040F809
2406000E
```

```
        8E020004 0040F809 24040032 0411FFB1 24040005 00409821 0000A021
40944800
        00008021 10000005 3C129D00 40104800 0270102B 1440001E 24050002
8E420020
        0040F809 8E240000 1440FFF8 24040064 10000025 00000000 40104800
0270102B
        14400013 24050002 8E420020 0040F809 8E240000 1040FFF8 24040020
1000001F
        00000000 40104800 0270102B 14400008 24050001 8E420020 0040F809
8E240000
        1440FFF8 00000000 1000001A 3C119D00 00052823 3C029D00 8C420080
00A02021
        0040F809 00052FC3 8FBF0024 8FB40020 8FB3001C 8FB20018 8FB10014
8FB00010
        03E00008 27BD0028 0411FF7A 00000000 00409821 1000FFDC 3C129D00
0411FF75
        00000000 00409821 40944800 1000FFE1 3C129D00 8E320064 8E220000
3C037735
        34639400 8C420000 0062001B 004001F4 00001012 72028002 24021696
0202001B
        004001F4 00002012 8E220080 0040F809 00002821 00408021 8E220080
2404000A
        0040F809 00002821 02002021 0240F809 00402821 1000FFD5 8FBF0024
        20555043 65657073 6F742064 6F6C206F 000A0D77
    End CFunction
```