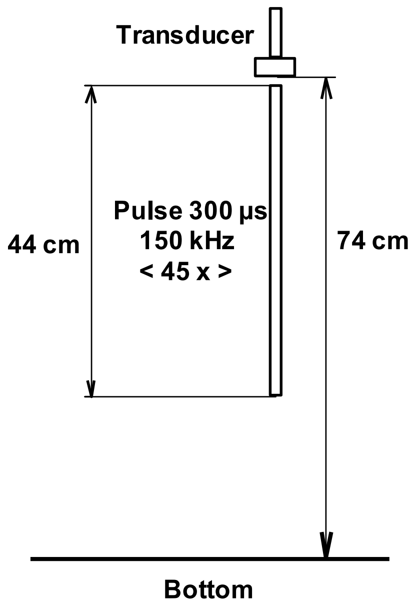
Mobile recording depth sounder

Purpose
Recording water depths with their position for a depth survey. The device must be suitable for mobile use in a dinghy without electrical installation.

Data sources
A depth sounder for the water depth and a GPS receiver for the position.

Display and storage
The depth is displayed digitally on a voltmeter. (limited to 5m)
The depth and position are recorded on a mini SD card in a txt. File. The data can be processed at a later stage.

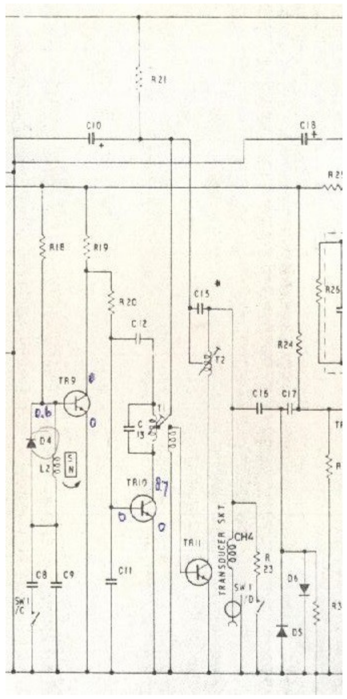
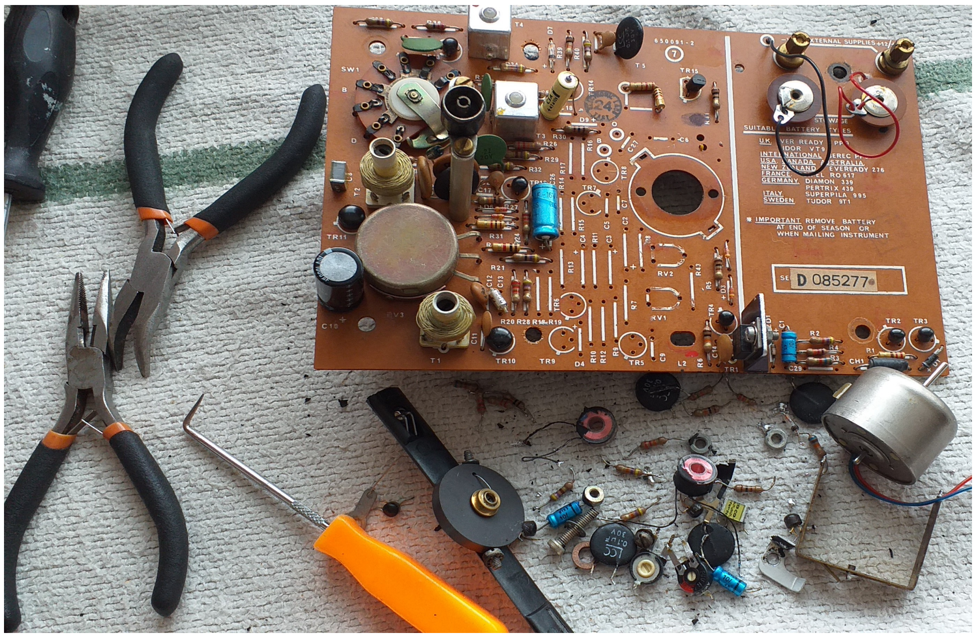
The sounder
For this project, the transmission and reception part of a Seafarer Mk3 or 4 Sounder is used.
The measured time of the transmitting pulse is approximately 300 µs. During that time, the pulse travels 44.4 cm.
(vSound = 1480 m/s) = (148000 cm/s) = (0.148 cm/µs) = (44.4 cm/ 300 µs)
A 150 kHz transducer is used that makes 45 vibrations during this pulse,
(150 kHz) = (150000 vibration/ s) = (0.150 vibration/ µs) = (45 vibrations / 300 µs)
Sketched somewhat to scale for a period of 1000 µs, this looks like this:



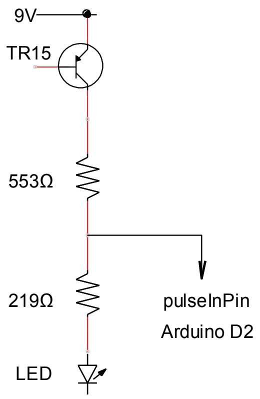
The smallest measurements are around 45 cm.

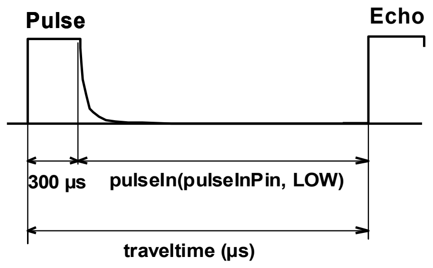
Originally, the depth of the Seafarer was made visible with an LED on a rotating arm that lit up when receiving the ultrasound. On a circular scale around the arm, the depth could be read.

Arduino Uno type microprocessor
Because the data has to be processed digitally, I use an Arduino Uno microprocessor to trigger the transmitter and to measure the duration. The motor, arm and other unnecessary parts have been removed from the printed circuit board to save power and to make room for Arduino and batteries.

The point between resistors R19 and 20 is the point where the Arduino triggers the transmitter on transistor TR10. R19 itself and TR9 have been removed. (The parts are also named on the print.)
The Arduino is programmed for this purpose via its USB connection. The program is retained on the Arduino and starts running as soon as there is voltage on the Arduino. With the function pulseIn(pulseInPin, LOW), the microprocessor calculates the length of time that the signal is 0. In other words, the time between transmitting and receiving pulse



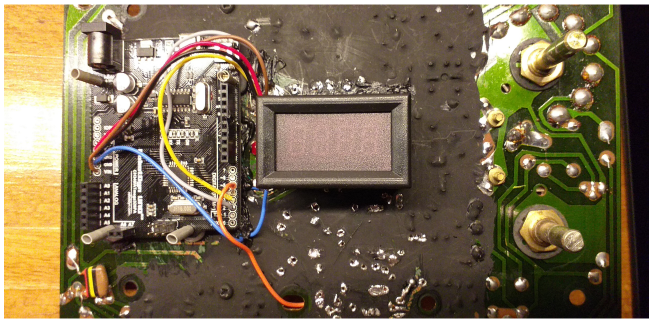
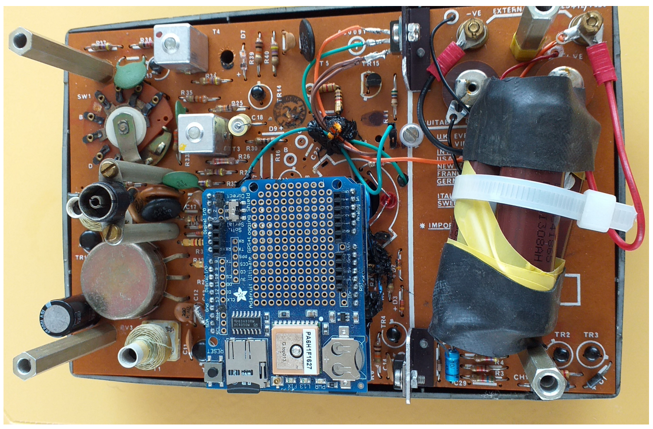


Depth in cm = travelTime x vSound / 20000.0

For a vSound in m/s and travelTime in µs, the Depth in cm is: travelTime x vSound / 20000.0
The speed of sound in water, vSound = 1480 m/s
An example: (The earler sketch)
Suppose the duration between transmitting and receiving pulse is 700 µs, and the pulse duration 300 µs then the total traveltime is 1000 µs and the depth 1000 x 1480 / 20000 = 74 cm

Display
To show the depth I use a digital voltmeter with a refresh rate of 200 ms. (A digitally controlled LCD screen was difficult to read and too slow) The calculated water depth is converted into a volt value by means of PWM on a digital output.

The range of the meter is therefore limited to 5 meters. (5V limit Arduino) For the calculated numerical value that is written on the SD card, this restriction does not apply.

The program tor the depth sounder part

```
// 230322_Sounder_Timer0_pulses
// Trigger a pulse 2 times a second via pin 5 (orange)
// Messure time between pulse and echo on pin 2 (grey)
// Display depth with PWM voltage via pin 3 (yellow)
// 1500 m/s = 1.5 m/millis or 1.500 mm/micro

#include <SoftwareSerial.h>
#include <SD.h>
SoftwareSerial mySerial(8,7);
#define chipSelect 10
File logfile;
#include <Adafruit_GPS.h>
#include <SPI.h>
Adafruit_GPS GPS(&mySerial);
#define LOG_FIXONLY false

const int pulseOutPin = 5;
const int displayPin = 3;
const int pulseInPin = 2; //
const int vSound = 1480; // m/s
const int pulseOutDuration = 20; //microseconds
const int timeBetweenPulsea = 500; // milliseconds
float volt = 0.0;
unsigned long previousMillis;  //violate

void Update(unsigned long currentMillis)
{
  if(currentMillis - previousMillis >= timeBetweenPulsea)
  {
    previousMillis = currentMillis;
    // send a pulse to the sounder
    digitalWrite(pulseOutPin, HIGH);
    delayMicroseconds(pulseOutDuration);
    digitalWrite(pulseOutPin,LOW);

    // messure depth and display
    unsigned long travelTime = pulseIn(pulseInPin, LOW) + 300;
    // 300 is pulselenght at mode 6x
    unsigned long depth = travelTime * vSound / 20000; //cm
    // devide depth in a way volts equals meters approximately
    volt = depth/ 1.55;
    if (volt >= 255) // max of range 5V
      volt = 255;
    analogWrite(displayPin,volt);

    // get $GPRMC from GPS
    if (GPS.newNMEAreceived())
    {
      char *stringptr = GPS.lastNMEA();
      if (!GPS.parse(stringptr))
        return;
      if (LOG_FIXONLY && !GPS.fix)
        return;
      //write the string to the SD file
      uint8_t stringsize = strlen(stringptr);
      if (stringsize != logfile.write((uint8_t *)stringptr, stringsize))
        Serial.println("ërror");
      if (strstr(stringptr, "RMC"))
      {
        unsigned long travelTime = pulseIn(2,LOW) + 300;
        unsigned long depth = travelTime * vSound / 20000;
        logfile.println();
        logfile.print("$SDDBT,");
        logfile.println(depth);
        logfile.println(millis());
        logfile.println();
        logfile.flush();
      }
    }
  }
}

void setup()
{
  pinMode (pulseOutPin, OUTPUT);
  pinMode (pulseInPin, INPUT);
  pinMode (displayPin, OUTPUT);
  pinMode(chipSelect, OUTPUT);
  if (!SD.begin(chipSelect))       // see if the card is present and can be initialized:
    Serial.println("Card init. failed!");
  char filename[15];
  strcpy(filename, "RMCDPT00.TXT");
  for (uint8_t i = 0; i < 100; i++)
  {
    filename[6] = '0' + i/10;
    filename[7] = '0' + i%10;
    // create if does not exist, do not open existing, write, sync after write
    if (! SD.exists(filename))
    {
      break;
    }
  }
  logfile = SD.open(filename, FILE_WRITE);

  GPS.begin(9600);
  GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCONLY);
  GPS.sendCommand(PMTK_SET_NMEA_UPDATE_5HZ);  // 5 times per second
  GPS.sendCommand(PMTK_API_SET_FIX_CTL_5HZ);
  GPS.sendCommand(PGCMD_NOANTENNA);
  OCR0A = 0xAF;
  TIMSK0 |= _BV(OCIE0A);
}

// Interrupt is called once a millisecond
SIGNAL(TIMER0_COMPA_vect)
{
  char c = GPS.read();
  #ifdef UDR0              //USART I/O Data Register
    if (c) UDR0 = c;
  #endif
  unsigned long currentMillis = millis();
  Update(currentMillis);
}
void loop()
{
}
```

The GPS position and the SD Card section

To receive the GPS position and to store the data, a second Arduino is used with a GPS shield on it. This contains a GPS receiver and an SD card holder. By using a second microprocessor, the timing of the depth sounder is not confused.
The first Arduino takes care of controlling the transmitter and indicates the calculated depth on the voltmeter. The second Aduino takes care of storing the GPS position and the calculated depth on an SD card.

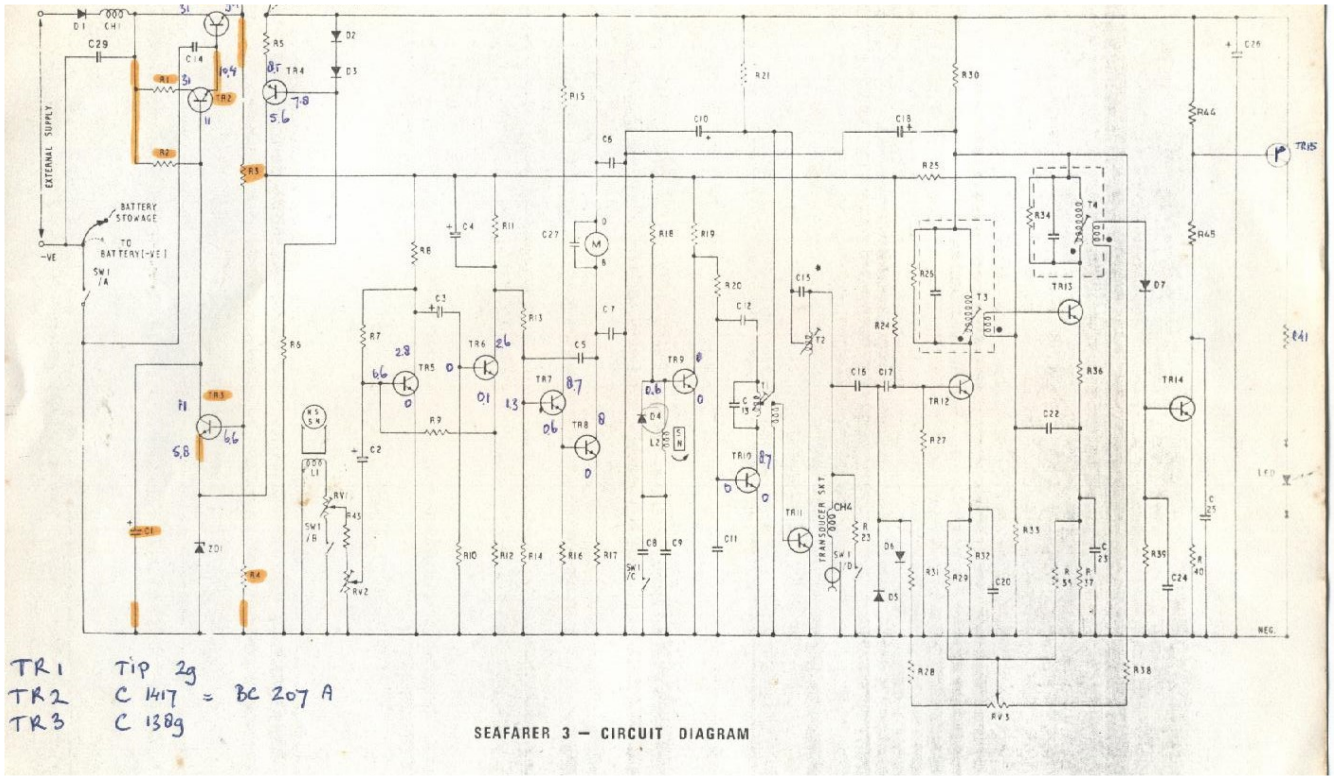
The program for the second Arduino

```
// 230322_shield_sdlog                    Saving $GPRMC string and depth on SDcard
#include <SPI.h>
#include <Adafruit_GPS.h>
#include <SoftwareSerial.h>
#include <SD.h>
#include <avr/sleep.h>
SoftwareSerial mySerial(8, 7);
Adafruit_GPS GPS(&mySerial);
#define chipSelect 10
File logfile;
const int vSound = 1480; // m/s
void setup()
{
  Serial.begin(115200);
  pinMode(chipSelect, OUTPUT);
  if (!SD.begin(chipSelect))              // see if the card is present
    Serial.println("Card failure");       // message via USB to serial monitor program
  char filename[15];
  strcpy(filename, "RMCDPT00.TXT"); // Automatic numbering of TXT files
  for (uint8_t i = 0; i < 100; i++)
  {
    filename[6] = '0' + i/10;
    filename[7] = '0' + i%10;
    // create if does not exist, do not open existing, write, sync after write
    if (! SD.exists(filename))
    {
      break;
    }
  }
  logfile = SD.open(filename, FILE_WRITE);
  GPS.begin(9600);                                 // Baudrate GPS
  GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCONLY);
  GPS.sendCommand(PMTK_SET_NMEA_UPDATE_5HZ);  // 5 times a second
  GPS.sendCommand(PMTK_API_SET_FIX_CTL_5HZ);
  GPS.sendCommand(PGCMD_NOANTENNA);
  OCR0A = 0xAF;                  // Output Compare Register A
  TIMSK0 |= _BV(OCIE0A);  // Timer/Counter Interrupt Mask Register
}
SIGNAL(TIMER0_COMPA_vect)              // Interrupt is called once a millisecond.
{                                                // writing direct to UDR0
  char c = GPS.read();
  #ifdef UDR0                                    //USART I/O Data Register
    if (c) UDR0 = c;
  #endif
}
```

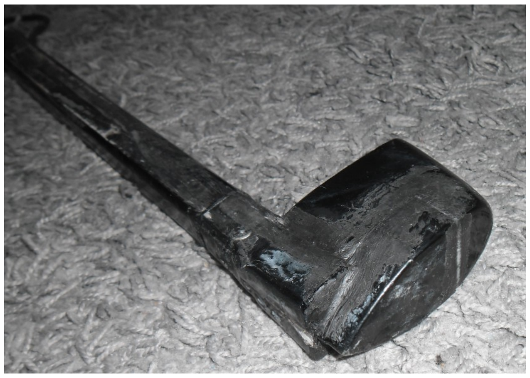
```
void loop()
{
 if (GPS.newNMEAreceived())
 {
   char *stringptr = GPS.lastNMEA();
   if (!GPS.parse(stringptr))
     return;
   uint8_t stringsize = strlen(stringptr);
   if (stringsize != logfile.write((uint8_t *)stringptr, stringsize))
      Serial.println("error");
   if (strstr(stringptr, "RMC"))
   {
     unsigned long travelTime = pulseIn(2,LOW) + 300.0;
     unsigned long depth = travelTime * vSound / 20000.0;
     logfile.println();
     logfile.print("$SDDBT,");
     logfile.println(depth); // (not NMEA standard, just DBT in cm)
     logfile.println();
     logfile.flush();
   }
 }
}
//POWERCONSUMPTION 90 - 100 mA
//(Depth sounder, Arduino's, GPS shield and Voltmeter display)
```

Example of a text file

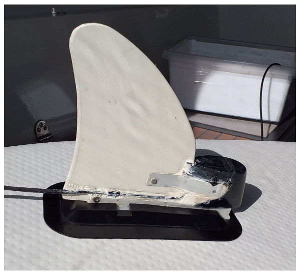
```
$GPRMC,102659.000,A,5254.2002,N,00537.5766,E,0.43,222.76,030917,,,D*6F
$SDDBT,64    // Not the standard NMEA Depth Below Transducer Sentence Format
             // but in cm in stead of meters with only one decomal
$GPRMC,102700.000,A,5254.2002,N,00537.5765,E,1.07,226.13,030917,,,D*67
$SDDBT,65
$GPRMC,102701.000,A,5254.1998,N,00537.5761,E,1.01,224.19,030917,,,D*65
$SDDBT,67
$GPRMC,102702.000,A,5254.1995,N,00537.5758,E,1.28,218.16,030917,,,D*6A
$SDDBT,63
$GPRMC,102703.000,A,5254.1992,N,00537.5754,E,1.33,217.97,030917,,,D*6C
$SDDBT,49
$GPRMC,102704.000,A,5254.1987,N,00537.5750,E,1.40,213.54,030917,,,D*64
$SDDBT,46
and so on....
```



SEAFARER 3 — CIRCUIT DIAGRAM

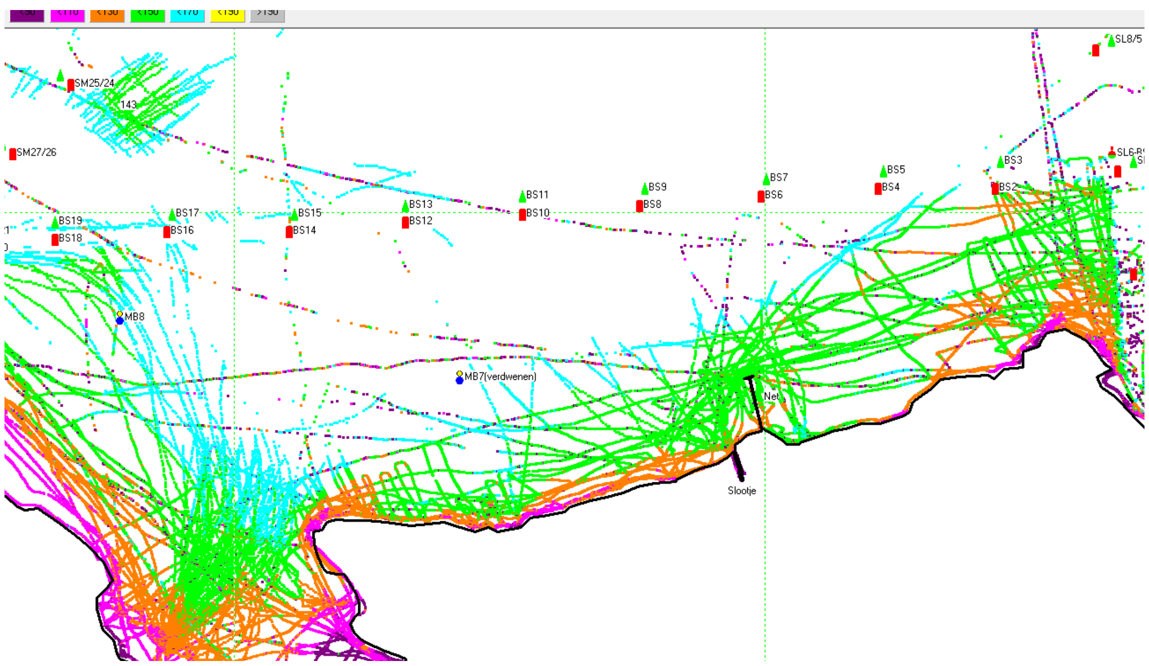


The 150 kHz transducer for use in a Europe dinghy. The stick and cable are pulled up from the outside into the front of the dagger board case. This causes the transducer, with fairing block, to lie against the hull. To the registered depth, 28 cm must be added in this position to get the water depth DBS (Depth below Surface). The sounder can be operated and read by the sailor.



The same transducer mounted together with a plastic fin on a nylon block which slides in the rails at the bottom of a SUP board. Approx 5 cm has to be added to get SDDBS

Example: Old recordings from 2017 projected on the buoyage situation of Oct 2021.



Jeroen Droogh bootprojecten@gmail.com