ARDUINO NANO BASED TELEPHONE CALLER LINE IDENTIFICATION INTERPRETATION SYSTEM

INTRODUCTION

This document describes the design and construction of a system based on an Arduino Nano (Atmega328P architecture) for interpreting and displaying details of incoming telephone calls to fixed telephone network subscribers. It is compatible with signaling which use a standard Bell 202 / V23 encoded call data record for call meta data. The meta data format is described in the ETSI specifications in the document list.

The core of the solution is an AFSK demodulator for the Atmel 8bit architecture written in C/C++ and does not rely on any purpose designed signal processing ICs. It is designed to use standard, easily available components. Additional software components includes a parser, display handler etc.

This solution is open and can be extended in ways which commercially available equivalent solutions do not allow. For example, the addition of large displays. Other developments could include white list based spam filters.

This project is not ideal for beginners because it will not be easy to get help if something goes wrong because there is a lot of dependencies on the local environment.

General disclaimer:

1. Before connecting anything, including this device, to your telephone network, be sure that it does not conflict with any local requirements or requirements of your telephone service provider regarding construction, materials, electrical safety, emission of electromagnetic radiation, certification or other such matters.
2. This development has been tested only in a Swiss network. Although the standards are in principle international, there are differences which may impact the correct functioning in other environments.

CONTENTS

ACKNOWLEGEMENTS

- The software AFSK demodulator is adapted from an APRS modem designed by markqvist (www.unsigned.io)
- Much has been liberally culled from various other Internet resources including the Arduino forums (http://forum.arduino.cc)

ABREVIATIONS

ADC	Analog to digital converter
APRS	Automatic Packet Reporting System
AFSK	Audio Frequency Shift Keying
CLIP .	Caller Line Information Presentation
Eeprom	Electrically erasable read only memory
ETSI	European Telecommunications Standards Institute
IC	Integrated Circuit
MCU	Microcontroller Unit
MDMF	Multiple Data Message Format
POTS	Plain Old Telephony System
SDMF	Single Data Message Format

FEATURE LIST

Display of incoming call information including number, timestamp and subscriber name when supplied by the telephone network.

Ring detection and display of ringing status.

External eeprom storage for lots of calls. This is currently set at 50 but that can be changed in the program. There is an option to select the internal eeprom with a reduced number of possible stored calls.

IR Remote controller for paging through and expanding the call information. The current display is a 1602 LCD (2 lines of 16 characters), so some scrolling is necessary to see all call details. Alternative hardwired buttons are also provided.

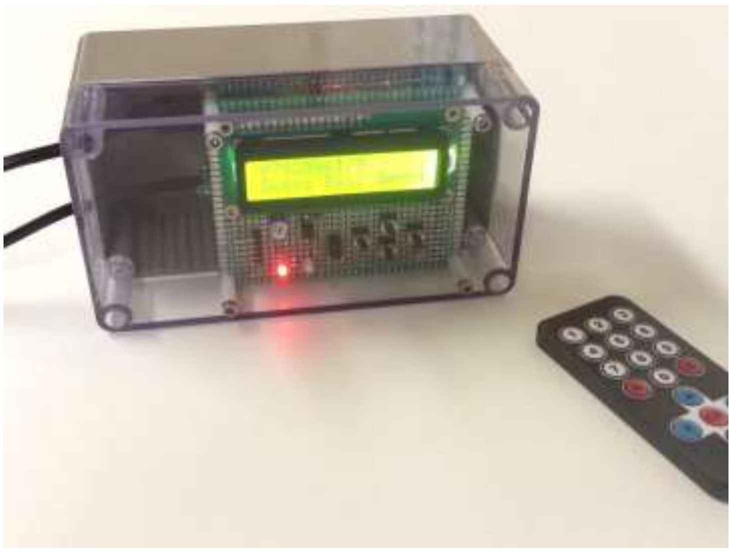
Display backlight intensity dependent on ambient lighting and with flicker suppression.

New call indicator

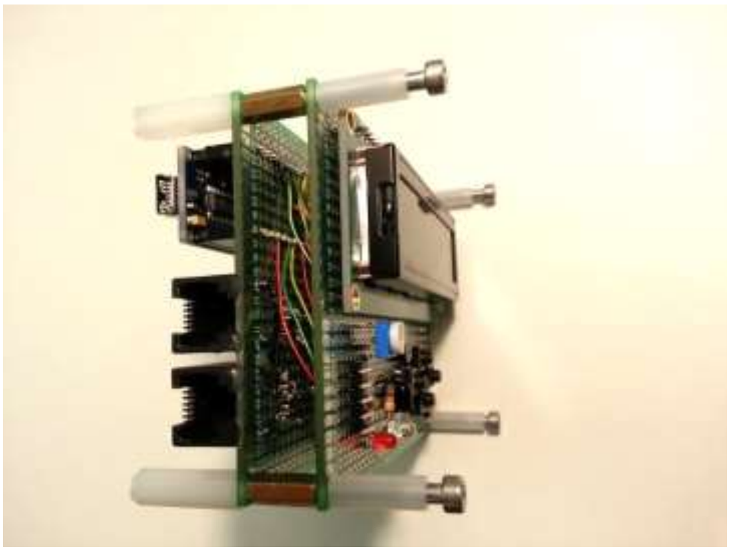
Software features to assist debugging of signal quality related problems.

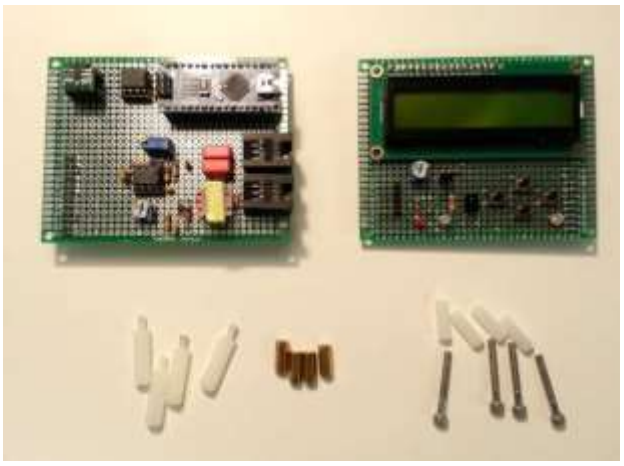
Note: currently only MDMF format CLIP is supported here.

PHYSICAL LAYOUT



Shown together with a remote control device for paging through the call history and expanding the entries.



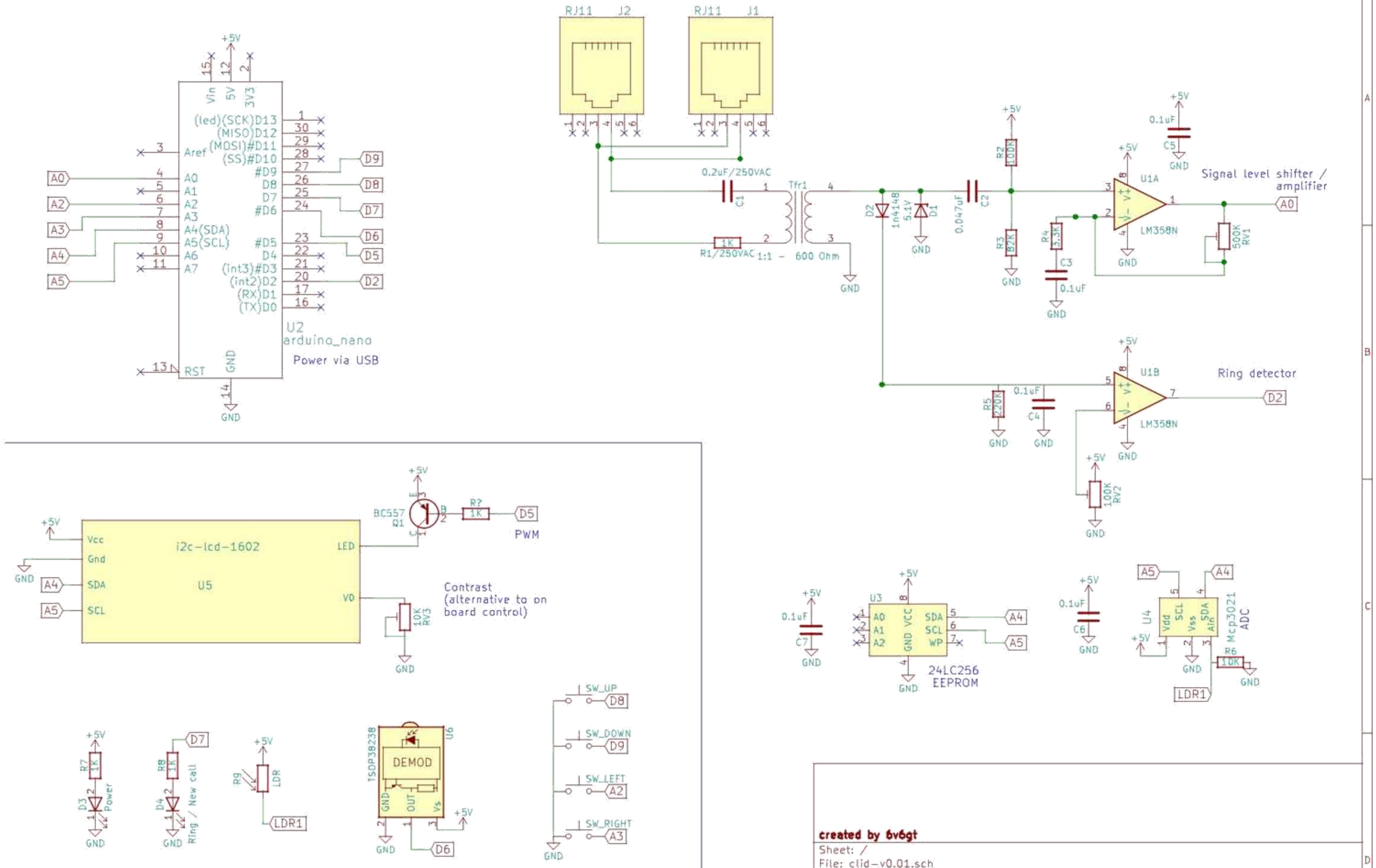


The two circuit boards are simply fastened together with screws. The electrical connections are made through matching header blocks which plug into each other to avoid direct wiring between the boards.

SCHEMATICS

1. MAIN BOARD(S)

1. The schematic represents both the demodulator board and the display board. In the solution illustrated, these are separate but it may even be easier to have everything on one physical board.
2. The potentiometer for the display contrast has been separated because the solder board has been sandwiched between the I2C backpack and the actual display and so it would not have been very accessible.
3. The only special remarks about the components are that the transformer is a 1:1, 600 ohm impedance miniature isolation type (eBay retailer) and the voltage rating of the network side components is that recommended in [Ref1].



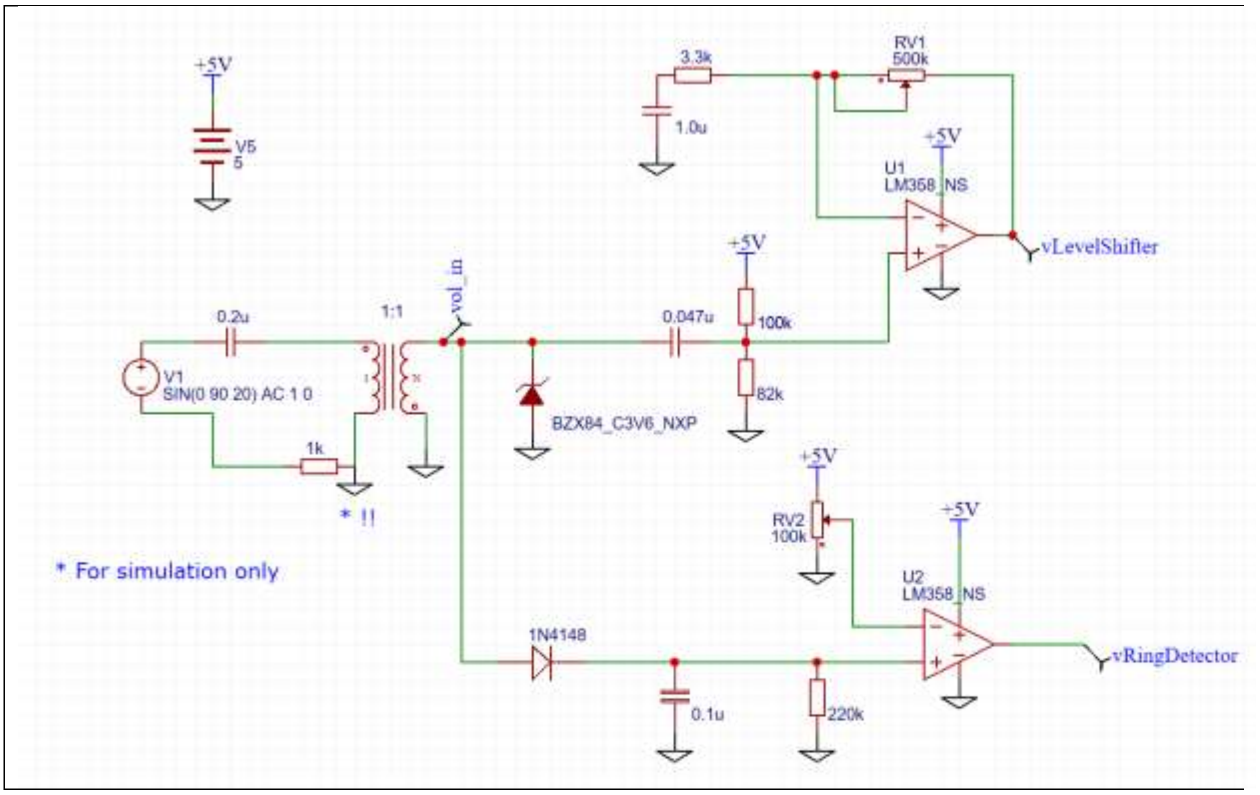
created by 6v6gt

Sheet: /
File: clid-v0.01.sch

Title: Caller Line Information Presentation System

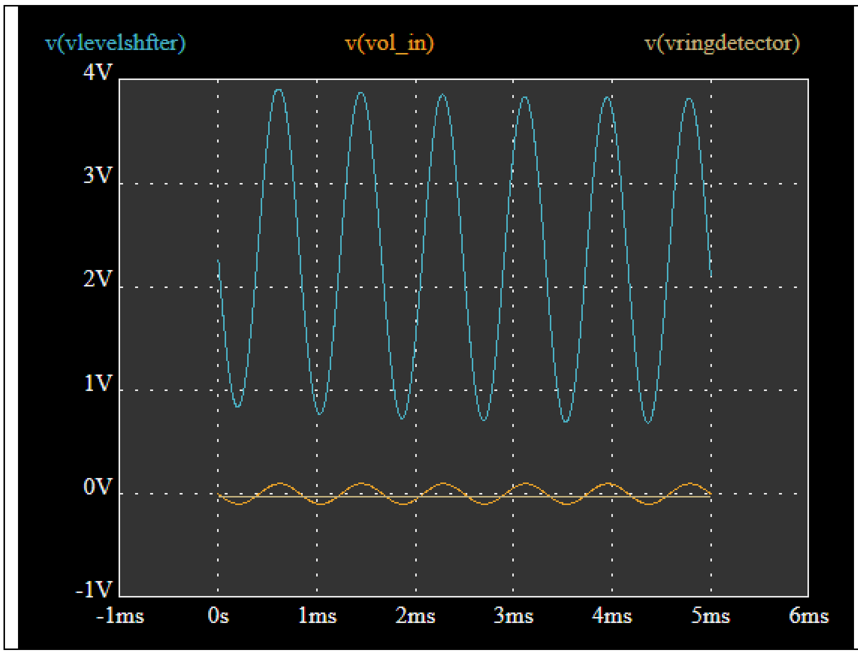
Size: A4 Date: 2017-07-19	**Rev: 1.00**
KiCad E.D.A. kicad 4.0.6	Id: 1/1

2. Simulation of Level Shifter and Ring Detector



To check the characteristics of the Level Shifter and Ring Detector I fed the circuits into EasyEDA with some typical inputs values from [Ref2]. The results appear below. Note that there is no correspondence between the component names used in the simulator and those in the main schematic diagram.

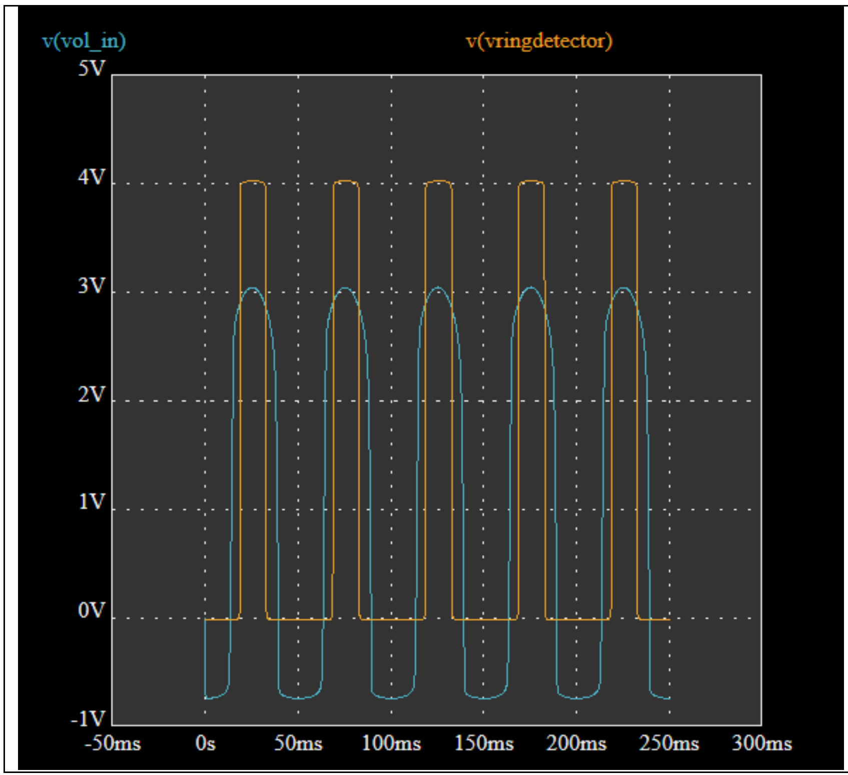
There are other minor differences also. The Zener diode in the simulation is 3.6 volt instead of 5.1 volts. Further, for the simulation it is necessary to earth both sides of the transformer, which of course you would not do in real life.



Level Shifter Simulation

parameters:

Frequency: 1200 Hz
VAC: 100mV
RV1 : about 50K effective
Simulation Step: 5uS
Stop after: 5mS



Ring Detector Simulation
parameters:

Frequency: 20 Hz
VAC: 90V
RV2 : 50:50
Simulation Step: 1mS
Stop after: 250mS

The level shifter probe has been removed to avoid unnecessary detail.

Some details of design decisions

Hardware

There is a separate ADC for ambient light intensity measurement to avoid developing a time sharing strategy for single ADC of the Atmega328p which is continually assigned to the demodulator. Other options also exist for handling the display backlight intensity such as leaving it on at normal brightness, dispensing with it or fully switching it off after a period of inactivity.

Alternative buttons and IR remote control unit have been implemented. Only one alternative is necessary and the IR Remote control is to be preferred because it is easier to expand the functionality, adding say codes for configuring the device such as #123* = backlight off etc. etc. The selected remote control unit supports 0-9, right, left, up, down, * and #

The buttons each use one digital pin. An alternative, if buttons are retained, is to use resistor chain and one analog pin. This would, however, also require a similar solution to the ambient light measurement because the Atmega328P on board ADC is dedicated to the demodulator.

The preamplifier for conditioning the signal is based on a design I found somewhere for a non-inverting level shifter with variable gain which I simulated it to determine the component values.

The ring detector is unusual because it simply (roughly) detects ring pulses at c. 20 Hz in a voltage window above that of the CLID signal. I did this to avoid further direct connections to the telephone service provider side of the circuit and minimise the requirement for additional high voltage tolerant components. The 20 Hz pulses are recognised in software.

The size of the data structures for storing call meta data were taken from the specification ETSI specifications [Ref7] and [Ref8]. These are quite large, for example specifying 50 characters for a subscriber name. Therefore, I have used an external 32Kbyte eeprom to hold a reasonable number (50) of calls. If only a small number of call records have to be stored, the internal eeprom can be used instead.

Software

The AFSK routine taken from the markqvist modem has, apart from the removal of the modulator, been left relatively untouched including comments. Some parts may now seem a bit heavyweight (eg the whole queue handling logic) which, in this reduced usage, now serves only as a 4 cell delay pipeline but it is elegant code so I have left it there instead of attempting to optimise it away.

SOFTWARE DISTRIBUTION

This is distributed as a zip archive name clid-v1.00 (initial version) because it consists of multiple individual components.

Location: http://forum.arduino.cc/index.php?topic=490392.0

Data formats

The best way of describing these is by example, using the output from the application in debug mode 1 which produces an 80 bit wide dump of the signal direct from the demodulator.

```
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000010101010101010101010101010101010101010101
01010101010101010101010101010101010101010101010101010101010101010101010101010101
01010101010101010101010101010101010101010101010101010101010101010101010101010101
01010101010101010101010101010101010101010101010101010101010101010101010101010101
01010101010101010111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111000000000110111001001010000000100001000010 0
00011001011101100101000110010111011001010001100100001110010110011001001101100100
10000001001010000100000110010111011001010011100100100110010000011001000011100100
+++++++10++++++++10++++++++10++++++++10++++++++10++++++++10++++++++10++++++++100
+++++++10++++++++10++++++++10++++++++10++++++++10++++++++10++++++++10++++++++100
+++++++10++++++++10++++++++10++++++++10++++++++10++++++++11111111111000000011000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
```

Training sequence of alternate 0 and 1

All Marks

0: start bit
1: stop bit

Data – little endian

Extraneous chars after check sum

The fields between the start(0) and stop(1) marks, some highlighted in purple are the actual call meta data and the first few are decoded below. The +++++++ pattern represents further characters in the data stream which have been obscured here. The parser is fed these bytes and produces the output for the display and storage in Eeprom.

Data Stream Value (little endian)	Hex Value / character	Description
00000001	0x80	Header byte to indicate MDMF data format
11100100	0x27	Length of following data packet in bytes
10000000	0x01	Date parameter type header marker
00010000	0x08	Length of date parameter in bytes
00001100	0x30	Ascii Digit 0 (first digit of month)
11101100	0x37	Ascii Digit 7 (second digit of month. 7 = July)
10001100	0x31	Ascii Digit 1 (first digit of day of month)
11101100	0x37	Ascii Digit 7 (second digit of day of month – so the 17th)
Etc. Etc.		

The encoding is described in several of the quoted references. [Ref5] and [Ref6] are the easiest to follow. The ETSI documents [Ref7] and [Ref8] are the most comprehensive.

TROUBLE SHOOTING TIPS.

1. Keep any mobile phone you are using for testing a few meters away from the circuit to avoid interference.
2. If you don't get a clean pattern in debug mode 1 when there is no activity on the line, i.e. no ringing, then you have to solve that before proceeding further. The variable resistor RV1 should have an initial setting of around 60k, but adjust as required. Debug is mode 1 selectable by entering a 1 in the serial console and 0 to revert to normal mode.
3. In standard mode, debugging information is written anyway to the serial console which can be useful if the signal delivered to the MCU is of a reasonable quality.
4. There is some fine tuning of the mark and space frequencies available to handle different network standards. See AFSK.h (source software)
5. I had some success using an audio recording of an AFSK signal captured from a telephone line, and feeding this into the demodulator via the PC sound card for testing.

A DOCUMENT COLLECTION, SOME DIRECTLY REFERRED TO ABOVE OR USED IN THE SOLUTION AND SOME GENERALLY RELEVANT FURTHER READING.

[Ref1] LibAPRS an Arduino soft modem library which provided the basis for the demodulator code.

http://unsigned.io/projects/libaprs/

[Ref2] Rane Note: Interfacing Audio and Pots including some basic electrical parameters of the telephone network and an interface circuit.

http://www.rane.com/note150.html

[Ref3] Randolf Telecom Inc. AN-4 (Midcom TN #98) LOW COST TELEPHONE LINE INTERFACE (DAA, FXO)

This is a comprehensive description including a specimen circuit for interfacing to POTS telephone networks

http://www.randolph-telecom.com/articles/AN-4,%20Low%20cost%20telephone%20line%20interface%20_DAA,%20FXO_.pdf

[Ref4] Cypress AN2336 - PSoC® 1 - Simplified FSK Detection

This is an example of using a Cypress manufactured chip for FSK signal processing. It is interesting here because it contains a comprehensive theoretical description of the processing techniques which are relevant to the AFSK demodulator in this solution.

http://www.cypress.com/documentation/application-notes/an2336-psoc-1-simplified-fsk-detection

[Ref5] holtek Type I caller ID using the HT9032

http://www.holtek.com.tw/documents/10179/116745/an0053e.pdf

[Ref6] EXAR TAN008 Designing Caller Identification Delivery Using XR-2211 For U.S.

https://www.exar.com/files/documents/tan_008.pdf

The two references above are similar application notes for obsolete ICs which describe the data formats and parsing of Caller Line Identification information. Exar has also a similar document relevant to British Telecom networks (TAN009)

ETSI documents

[Ref7] **ETSI EN 300 659-3 V1.3.1**
http://www.etsi.org/deliver/etsi_en/300600_300699/30065903/01.03.01_40/en_30065903v010301o.pdf

[Ref8] **ETSI 2 EN 300 659-1 V1.3.1**
http://www.etsi.org/deliver/etsi_en/300600_300699/30065901/01.03.01_60/en_30065901v010301p.pdf

These are very comprehensive documents describing and specifying relevant telephony data formats and encoding standards at various layers.

Suggestions for optimisation and future development.

Each call currently uses 81 bytes of external eeprom. With a 32Kb eeprom, this is no big problem but if you want to store 1000s of calls, the structure can be optimized for example, by reducing the length of the subscriber name field from the specified 50 characters to say 20 characters.

Count of repeated call attempts by the same caller in the same day to avoid storing a separate record per call.

Expand the parser to handle other data format types such as SMDF and additional parameters which may relevant for other telephone networks. (I can't test these live – only by simulation)

Add year information to supplement the time stamp in the call data record.

Add a white list of known callers to the eeprom for personalised name display.

Again a white list for possible spam rejection.

Possible ultimate spam handling strategy. Always drop the first ring burst. On calls from known callers allow subsequent rings through, reinserting the call data record between the second and third ring burst so the CLID appears on compatible phones. (Requires also an AFSK modulator to regenerate the call data record and the device must be in circuit before any phone). Unknown numbers fall silently onto a network voice mail service.

And finally, code can always be tidied up.

And also finally, hardware can always be tidied up here especially so with a PCB instead of prototype boards and a nicer layout maybe on 1 board instead of spread over 2 boards.