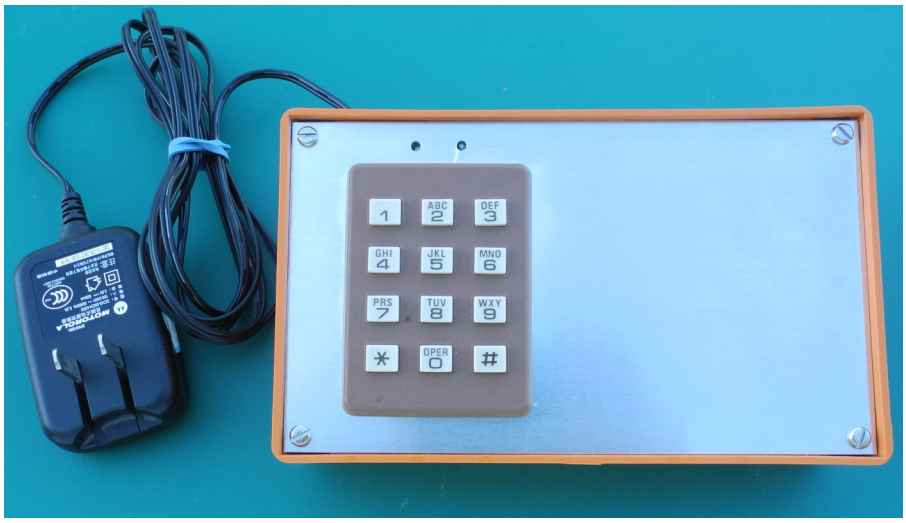
DTMF Autodialer Project



The DTMF autodialer project is based on a HOLTEK HT9200A tone generator chip controlled by an Arduino Nano micro-controller board. The parts are readily available from internet stores at reasonable cost. The autodialer user interface is very simple, consisting of a standard telephone type keypad and two indicator LEDs. The project is easy to build and the executable file (Intel hex format) is provided for uploading to the micro-controller's flash memory. The project source code files are included.

Date: 2016 October 1

Table of Contents

1 Introduction

This project describes a simple DTMF autodialer that stores up to 41 telephone numbers, each stored number can be up to 20 digits long. Operation of the autodialer is extremely simple, it takes two or three key presses to dial any of the stored phone numbers. The autodialer is completely stand-alone and connects in parallel with the subscriber line on any telephone unit. The autodialer is connected to the telephone line only when a number is being dialed, otherwise it is completely disconnected by an internal relay.

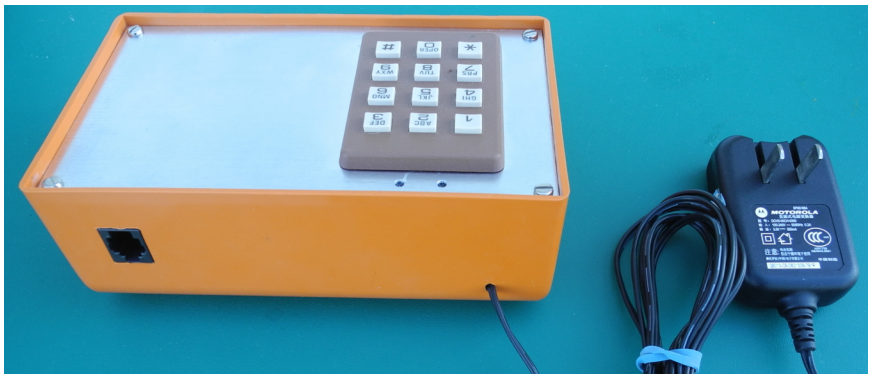
The autodialer is designed around a Holetk HT9200A dtmf tone generator chip and an Arduino Nano micro-controller board. It uses a standard telephone type keypad with 12 keys for number entry and dialer control. Two small LEDs provide indication for the dialer operating modes, a blue LED for normal dialing mode and a red LED for number entry mode.

The parts list describes all the required components for the autodialer project. Most of the parts used for this project are readily available from numerllous sellers on eBay and other on-line retailers. The prototype uses SMT devices as well as standard leaded devices, the choice of devices being made soley on what components were on hand at build time. SMT devices are plentiful these days and the small size uses less space, but handling such small components requires a steady hand, good eyesight and a practiced soldering technique.

The software for the autodialer project is provided as an executable file (holtek-autodialer.hex), ready for uploading to the ATmega328P chip on the Nano controller board. Program source code is included. A method for programming the ATmega328P chip using a USB ISP programmer and the AVRDude software tool is described in the documentation.

The Arduino Nano board is used for this project because of its small footprint, low cost, on-board 5 V supply, and 6-pin ISP header programming interface. All of these features are used in the autodialer project.

A rear view picture of the autodialer is shown below. Here you can see the standard RJ-11 phone jack for suscriber line connection.



The finished autodialer in a Hammond plastic instrument case.

The autodialer program is written in Atmel AVR assembly language using the Atmel Studio 7 IDE development tool available free from the Atmel website. The source code files are commented and it should be fairly easy to read and understand the program.

You may wonder why the bother to build a dedicated autodialer when most modern phones already include a user directory for storing frequently called numbers. One reason for a DIY autodialer is that most phone directories are a pain to use, needing several key entries to dial out the desired phone number. These dialers are built in with the phone and so cannot be used with other phones. The autodialer project on the other hand is very easy to set-up and use, it is a speed demon, and it can be used with any phone, even the ones that come with their own built-in user directory dialing facility.

You may find this autodialer useful if you need a dedicated dtmf keypad for your application, say for remote signalling and control, or for a security entry control system, and you have to write your own keypad control program. The autodialer program can be used as the basis for your own controller program and it has lots of routines that can be reused to save on development and testing time.

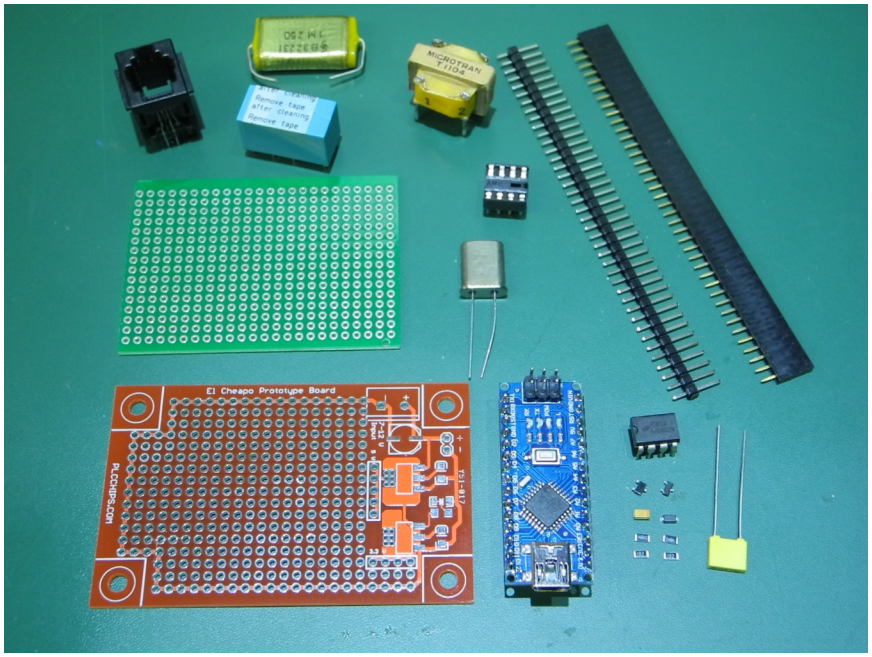
If you are looking for a simple, useful project that works well and is easy to construct, this autodialer may fit the bill. With the full source code provided, you can customise the operation any way you wish. You can devote more time building the autodialer and not worry that it may not work after putting in so many hours into the build. We hope you enjoy building and using the autodialer as much as we had in designing, testing and building the prototype device.

2 Parts list

You need the following parts for the autodialer project:

1. Holtek HT9200A dtmf tone generator chip in 8-pin DIP package. The chip runs on a Vdd of +5 V.
2. 3.58 MHz crystal, type used for TV color-burst frequency.
3. Arduino Nano controller board, 16 MHz, with ATmega328P microcontroller chip.
4. 5 V power supply (a 5V cell phone charger connected to the +5 V pin will work). A Motorola cell phone charger rated at 5.0 V at 550 mA is used in the prototype. You can also use +7 V to +12 V (recommended) external power supplied to the VIN pin to power the autodialer.
5. 1 ea. 5 V DIP relay.
6. Miscellaneous hardware:
    1. Two small pieces of perforated prototyping board (main circuit board and DAA board).
    2. Telephone style keypad, 3 columns by 4 rows.
    3. Two LEDs, red and blue. Any two colours will do.
    4. Two 2-pin headers (DTMF output and relay driver output), one 3-pin header (for LEDs), one 7-pin strip header socket (for keypad).
    5. Small telephone isolating transformer. Best to use reclaimed transformer from junk telephone equipment, modems, etc. The prototype uses a Microtran T2104 transformer. A smaller T1104 can also be used.
    6. RJ-11 style telephone jack.
    7. 1 ea. 2N2222 Si NPN transistor. Not critical, any small signal device is suitable.
    8. 1 ea. 2N7000 N channel FET, for relay driver.
    9. 1 ea. 2N4001 diode.
    10. Resistors per schematic. Both SMT types and axial lead types used, depends on what you have on hand.
    11. Capacitors per schematic. 47 uF/16 V tantalum or electrolytic capacitor for power supply bypass. Types not critical. For DAA isolation capacitor, use a capacitor with at least a 250 Vdc rating.
    12. Wire-wrap wire (30 AWG), plastic project case, plastic screws & nuts, spacers etc.
    13. 1 ea. transient absorber, 230 Vdc.

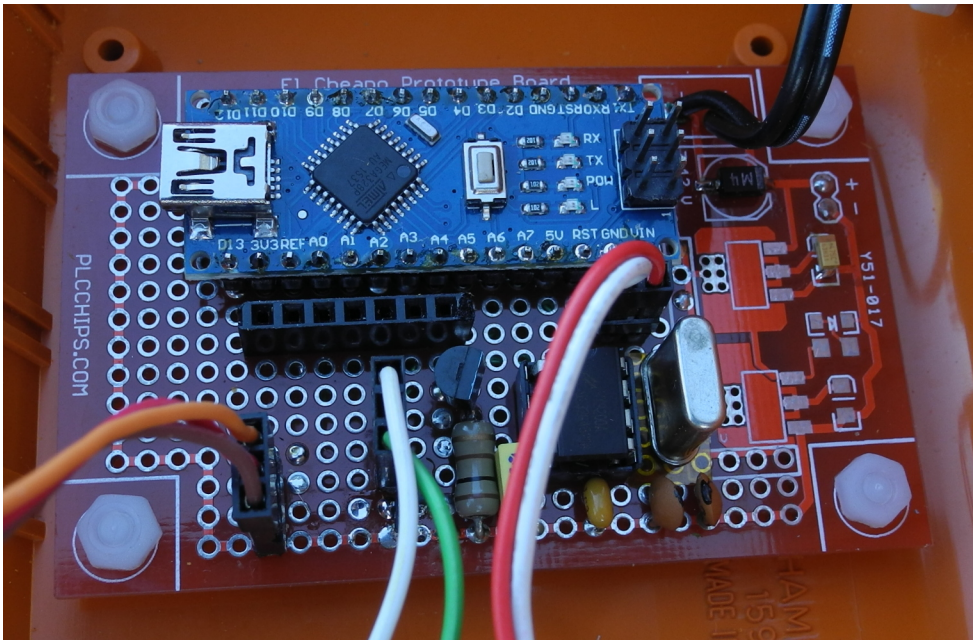
14. 1 ea. Mini Desk COnsole, Hammond no. 1595, approximately 6.4L x 3.75D x 2.25H inches, with sloping aluminum front panel. This case has ample space for all the parts used, but any type of case can be used. The picture below shows some of the main components used. Note the SMT devices below the HT2900A IC and how small they are.



3 Construction details

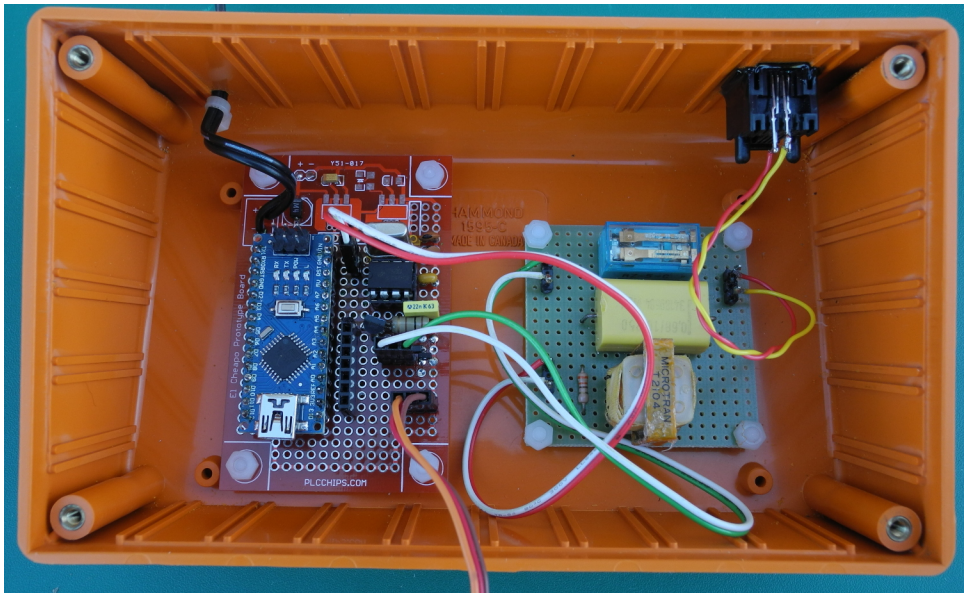
The autodialer is built on two boards, a main board and a data access arrangement (DAA) board. The picture below shows a close-up view of the prototype main board layout, top view. The main board has the Nano controller board, the HT2900A chip in a DIP socket, the 3.58 MHz crystal, the 2N7000 FET relay driver and the 2N2222 NPN transistor dtmf signal driver. The required resistors and capacitors are also mounted on this board. Place a piece of Kapton insulating tape between the base of the crystal and the circuit board to prevent any possibility of short circuits.

Connections to off-board devices are made via a 2-pin strip header and two 3-pin strip headers. A 7-pin mating strip header connector is used to connect the 3x4 keypad to the main board. The two black wires on the top right are from the external cell phone charger. The red and white wire pair goes to the Microtran transformer. The green and white wire pair (3-pin header) goes to the DIP relay. The brown, red, orange wires go to the LEDs on the front panel.



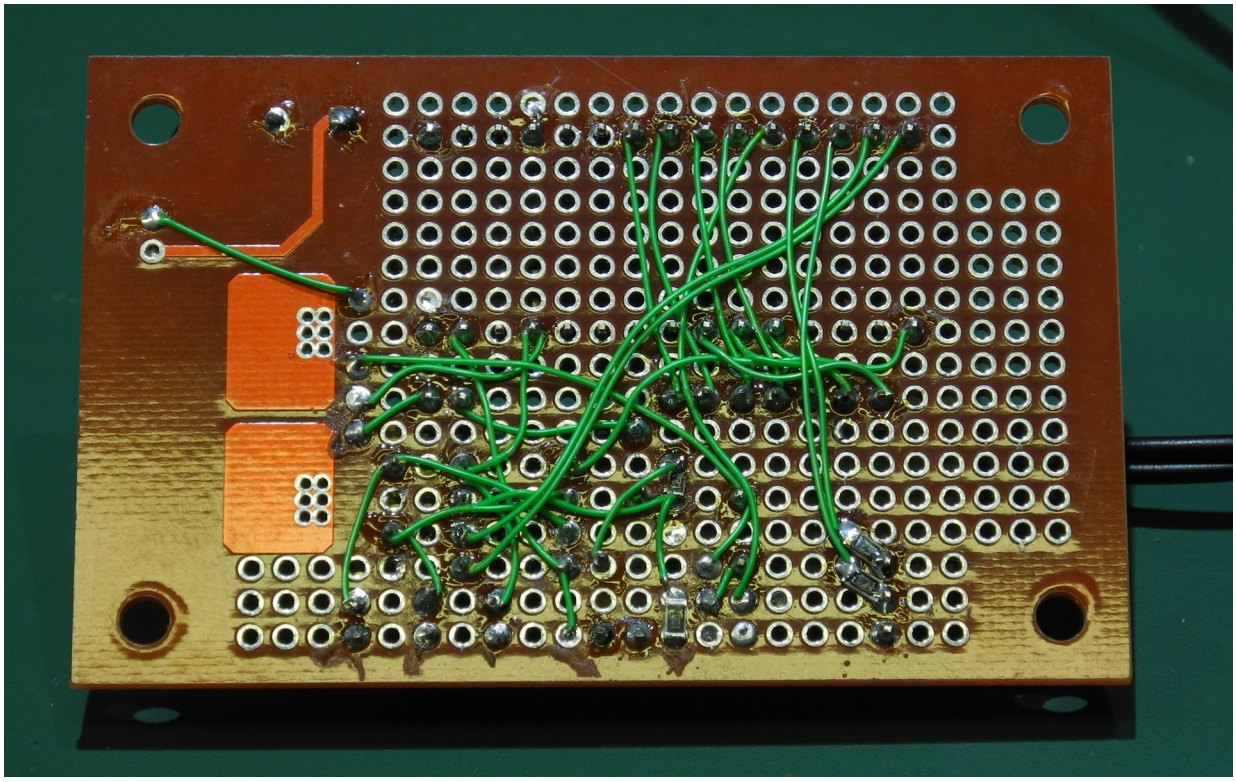
For the main board of the prototype a custom prototype PCB is used but you can use any small piece of perforated prototyping board, like

the green garden variety board show in the parts picture and used for the DAA board. A double sided board, one with solder pads arranged on a 0.1 x 0.1 inch grid pattern on both sides and with plated through holes is preferred. The next picture shows an inside view of the finished autodialer. You can see the DAA board arrangement in this picture.

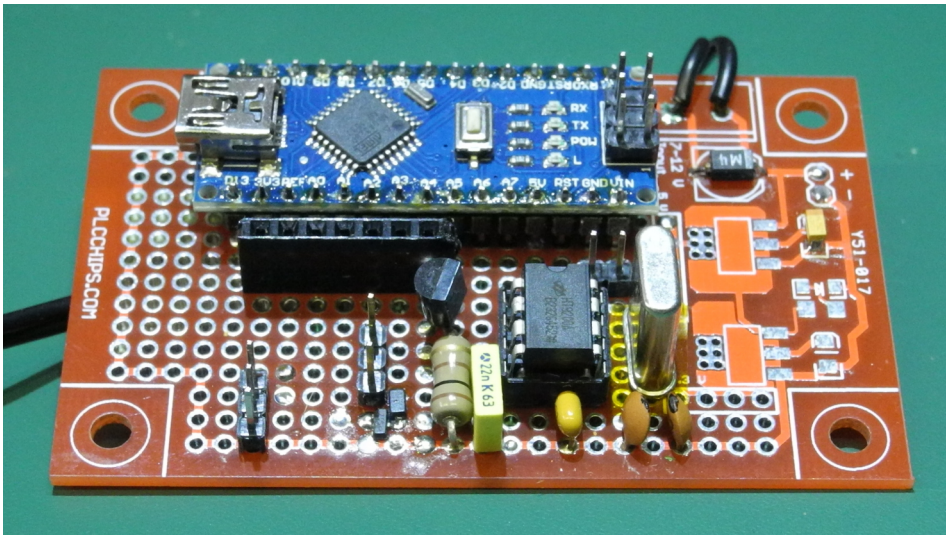


The main board and DAA boards are wired on the underside with insulated wire. 30 AWG Kynar insulated wire-wrap wire is used for the wiring and it is relatively easy to make the wire connections if you use a wire-wrap tool with a built-in stripper. Solder each end of the wire as you make the interconnections. The picture below shows what the prototypt main board wiring looks like.

Work slowly and carefully when doing the wiring. Check each wire connection you make against the schematic diagram and make sure you are wiring the correct pins and components. The wiring phase is the most likely time for making mistakes, so take extra time to do this work. Look especially at the +5 V and +3.3 V power and ground wiring as this type of check-out is one of the easient ways to avoid experiencing a “smoke” event.



Main board wiring, back view

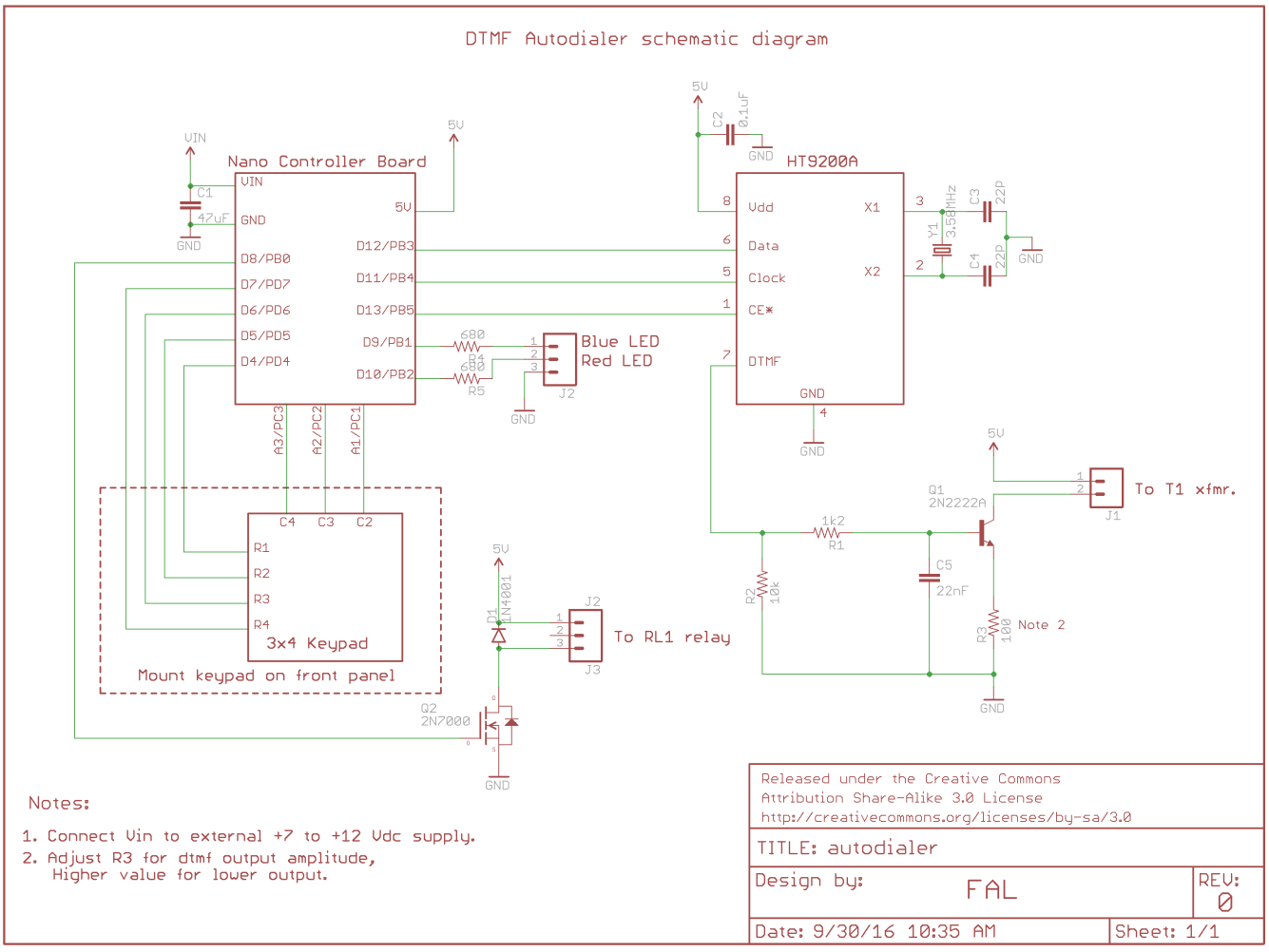


Main board, top view

4 DTMF autodialer circuit description

Refer to the autodialer schematic diagram for circuit details. The Nano controller board interfaces to the HT2900A dtmf generator chip via an SPI serial interface. Please refer to the Arduino Nano schematic diagram and documentation for details on the micro-controller board. A lot of information on this device is available from the Arduino and other websites.

Please refer to the Holtek data sheet for specific details on the HT2900A dtmf generator chip.



4.1 The HT9200A dtmf generator operation

The HT9200A device uses a 3.58 MHz crystal as a reference for generatinhg accurate dtmf signals. An on-chip oscillator drives the crystal. The HT9200A has integrated crystal load capicitors, but you may need to add external capacitors if the crystal's recommended loading capacitance is higher. The prototype design uses additional 22 PF capacitors.

The device is controlled by an SPI (serial peripheral interface) so it needs only a Clock and Data lines for communications with the Nano controller. The Clock line is connected to port pin D11, the Data line to port pin D12. The dtmf chip operates as a slave device.

An active low chip-enable (CE*) input controls the activation of the device and is driven by the D13 port pin.

The HT9200A dtmf output line DTMF drives a 2N2222 buffer transistor Q1, which in turns drives isolating transformer T1 on the DAA board. R1, R2 and C5 are for biasing and filtering. R3 sets the gain of the buffer stage. The value of R3 can be changed to adjust the gain and thus the output signal amplitude. A higher value of R3 reduces the gain. R3 values should be in the range of 100 to 150 Ohms.

4.2 Nano controller interfaces

The Nano controller uses port pins A3, A2, A1 to scan the keypad column lines 4, 3 and 2 by sequentially driving each line in turn low. Column 1 is not implemented on the keypad. Port pin A0 for column 1 output scanning is implemented in the control program but A0 is not connected. The program is written to scan a 4x4 keypad with 16 keys, but only three keypad columns are connected on the autodialer.

Port pins D4 through D8 are used as input scan return lines for the keypad rows 1 through 4.

Port lines D9 and D10 are the blue and red LED outputs. R5 and R6 current limiting resistors installed on the main board to allow direct connection to the LEDs mounted on the front panel.

The Gate terminal of the 2N7000 MOSFET Q2 is connected to port pin

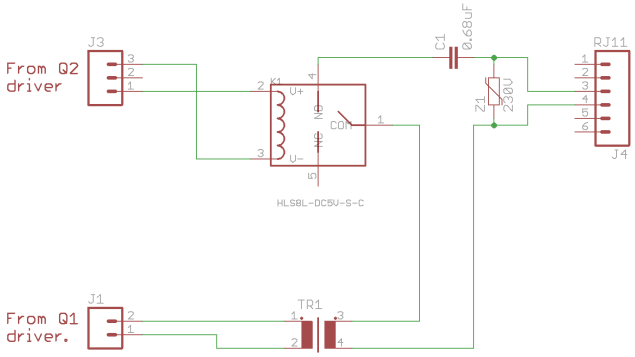
D8. The Drain terminal drives the HOOK relay RL1 that is mounted on the DAA board. This relay is used to connect the dialer's dtmf tone output signal on transformer T1's secondary winding to the suscriber line. RL1 is energized only during the time the dialer is transmitting telephone number tones.

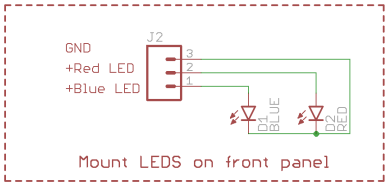
4.3 The DAA interface

The Telephone company requirements for connecting suscriber equipment to their lines is that all connections must be through an approved data access arrangement (DAA). The autodialer design uses a signal isolating transformer, a NO relay contact, and a DC blocking capacitor to achieve a high level of isolation. The DAA schematic diagram below shows the DAA details.

A transient absorber Z1 rated at 230 Vdc should be connected across the incoming phone line connections on the RJ-11 jack to protect against high induced fault voltages. The schematic diagram below provides details of the DAA interface.

DAA schematic diagram





Notes:

1. C1 is 250Vdc rated mylar.
2. TR1 is Microtran T1104/T2104 or similar

TITLE: daa-schematic		
Design by: FAL		REV: 0
Date: 9/30/16 3:53 PM	Sheet: 1/1	

5 Control Program description

Please refer to the holtek-autodialer(testing).asm source code file for details of the autodialer assembly language program. The program was developed using the Atmel Studio 7 IDE software suite and is written in assembly language. You can use Studio 7 to edit, assemble and link the autodialer program to generate a new executable hex file for flashing into the ATmega328P micro-controller.

There are two other associated assembler files, main.asm and eeprom.asm that have to be accessible for assembling the program. The main.asm file is the “main” file. We use main.asm as a holder that simply “includes” the real application file, holtek-autodialer(testing).asm. The eeprom.asm is “included” from holtek-autodialer(testing).asm. It has a set of standard eeprom sub-routines that are used in many different application programs, so it is kept in a separate file for easy maintenance.

Before we dive into the autodialer program details, a brief overview of the operating modes will lay the framework for understanding the program details.

5.1 Autodialer operating modes

The autodialer has two operating modes, the RUN mode and the PROGRAM mode. The autodialer starts up in the RUN mode. While in the RUN mode, the blue status LED turns on. When the autodialer is switched to the PROGRAM mode, the red status LED turns on. Special key sequences are used to toggle between the RUN and PROGRAM modes and to enter index numbers.

99#	Toggle between RUN and PROGRAM modes
nn#	Enter the one or two digit index number nn
*	Cancel the current operation

When the autodialer is started up for the first time the EEPROM is in an un-initialized state. The first thing to do is to program in some phone numbers. Enter the PROGRAM mode by pressing 99#. The blue LED turns of and the red LED turns on to indicate the autodialer is now in the PROGRAM mode.

5.1.1 The Autodialer PROGRAM mode

To program in a new telephone number first enter a two digit index number. For example, to enter a number into index location 10, enter 10. Notice the red LED flashes after each keypress to confirm a new key is entered.

Now enter the desired telephone number. For example enter 19056438793. When the telephone number entry is completed, press the # key to save at index location 10 in EEPROM memory the number 19056438793.

If you make a mistake while entering either the index number or the telephone press the * key to cancel the whole operation and start the operation over again.

In this manner, enter the rest of the telephone numbers you wish to store. You don't need to fill up the entire memory capacity of 41 numbers, you can always return to the PROGRAM mode and add or re-enter telephone numbers.

If you enter a two digit index number then press the # key, the indexed EEPROM slot for the telephone number is filled with 0xff's, which has the effect of erasing the 20 digit space. You don't need to erase a slot before writing a new telephone number to it, you just need to re-enter the new number and it will overwrite old data in the slot.

Return to the RUN mode once again by entering 99#. The blue LED turns on again to show the autodialer is back in the RUN mode.

5.1.2 The Autodialer RUN mode

Dialing a stored telephone number is done only from the RUN mode. To dial a number, first enter a desired index number and pressing the # key. Up to 41 indeces are accepted (00 through 40). The autodialer will also accept a 1 digit index number for indeces 0 through 9. to enter indeces 10 through 40 you must of course enter both digits. For example, to dial the telephone number stored at index 09, press 9#.

You may also enter 09# as a valid index number.

If you enter the wrong index number by mistake, press the * key to cancel the input and start over.

We can now get into some of the programming details in the next section below.

5.2 Program features and details

The holtek-autodialer program is structured in as modular a fashion as possible. Many of the routines are general purpose in nature and are used in many other applications. The description here will focus mainly on the autodialer specific code and will pass over the general purpose routines, such as the low level video driver routines and USART initialisation routine.

Some of the general purpose routines are included in the program but not used by the autodialer program. For example the sub-routines <pahex:> and <pdec:> are included but are not called. The reason is these may or may not be used in the development and debugging phase of the program and are left in the code for possible future use.

The hardware USART module is initialized by the program and a receive interrupt handler is provided. User access to the console input and output function is via sub-routines <ci:>, <getc:> and <co:>.

The autodialer program has a small section of code in <Main:> that print out a signon banner message to a console screen via the USB serial port. This signon message display is entirely optional and can be deleted without affecting the autodialer operation. It was included in the development code for convenience to indicate the serial communications routines were working. A lot of the temporary debugging test code relies heavily on dumping data to the console during testing of the various data buffer storage and retrieval routines.

The USART is set up for 19,200 baud, 8 data bits, 1 stop bit, no parity and no hardware or software flow control.

5.2.1 Interrupt handlers

The holtek-autodialler(testing).asm program uses two hardware timer-counter modules TCNT0 and TCNT1 for handling repetitive timing tasks. Both timers operate in the timer output compare mode. TCNT0 is set up to cause an interrupt every 4.096 ms, and TCNT1 is set up for an interrupt cycle every 32 us.

TCNT0 interrupt handler services a number of software timers, each time has associated timer flags for foreground program use. The keypad scanner service routine is also part of the timer interrupt handler.

The TCNT1 interrupt handler's main task is to handle the SPI interface on a regular basis, resulting in a SPI communications rate of 31.25 kHz. In this interrupt routine a polled SPI software handler is used rather than the SPI hardware module because of the non-standard 5 bit data size of the HT9200A chip. The user interface to the SPI transmitter consists of the <spiwr:> and software flag spirdyb.

The USART0 interrupt handler supports a buffered receive complete scheme for data arriving on the RxD pin. Incoming data is stored in a circular buffer SINB with head and tail pointers putcnt and getcnt. There is no interrupt handler for the TxD transmitter, serial transmissions are handled by polling.

5.2.2 RUN mode routine

Following a power-up or reset condition the controller initializes various hardware modules, prints a signon banner, then enters the RUN mode code at <runm0:>. The program waits for a keypad entry and checks any input to see if it is a * (cancel), #(execute) or other number n(index number).

One or 2 digit index numbers are checked for validity. Numbers must be in range 00 through 40, or the special 99 index. If the entry is 99, the program jumps to the PROGRAM mode code at <prgm0:>

If the index entered is in range of 00 through 40, <runm5:> reads the telephone number from EEPROM memory. The index number is used as an offset from the base address of the stored telephone numbers to fetch the correct telephone number. Sub-routine <gidx:> places valid index numbers in buffer indexb.

Code section <runm5:> converts the packed BCD index number back to binary format using the sub-routine <a2bn:>. The resulting binary index number is used as an offset to the base address of the stored telephone numbers in EEPROM, to sequentially read back the indexed telephone number in code section <runm6:>. For efficient use of the EEPROM memory space, the telephone numbers are stored in packed binary coded decimal (BCD) format, so each data byte holds two decimal digits.

Code sections <runm7:> through <runm9:> handle the control of the HOOK relay to connect the dtmf output to the suscriber lines, provide required time delays, and to transmit the dtmf control code to the HT9200A device.

The sub-routine <dial:> handles the actual communications to the HT9200A chip.

5.2.3 Program Mode routine

The PROGRAM mode is entered from the RUN mode by pressing 99# keys. The blue LED goes off and the red LED turns on to indicate the PROGRAM mode is active. Pressing 99# sequence again will return the autodialer to the RUN mode, the red led will go off and the blue LED will go on.

The PROGRAM mode waits for a two digit index entry plus whatever telephone number you wish to enter, up to a maximum of 20 digits. Pressing the * key cancels any entry sequence and returns to the start of a new entry.

Press the # key at the end of the telephone number to store the number in EEPROM at the entered index number. You must enter a full two digits for all indeces (00 through 40).

For example, to store telephone number 123-456-7890 at index position

02 your would press: 021234567890#

The code at <prgm0:> sets up the data buffers and control flags. Code from <prgm1:> through <prgm5:> checks for *, #, or 99# and takes the appropriate actions described above.

The code at <prgm3:> processes number key entries. The <gnum:> code section packs incoming BCD number pairs into a register buffer inbuf. Assembled and packed BCD digit pairs in inbuf are then stored in another multibyte RAM buffer tibuff.

The code section <prgm6:> calls <store:> sub-routine to store the assembled telephone number in tibuff to EEPROM memory at an offset location specified by the entered index number.

6 Programming the micro-controller

After the autodialer assembly is completed and checked again for possible wiring errors, it is time to program the executable holtek-autodialer.hex file you downloaded from the PLCCHIPS.com web site, to the Nano's flash memory.

Connect the DAA board to the main board and the LEDs and keyboard to the main board. You can safely program the Nano with everything connected.

During the programming procedure, power is supplied to the Nano via the +5 V Vcc connection on the ISP programmer.

6.1 Burning the holtek-autodialer.hex program.

The programming method used to upload the executable hex file to the Nano's flash memory uses the AVRDude programming software and a USB-ISP-programmer avaialable from many on-line retailers. The Arduino IDE is not used in this method of programming, nor is the boot-loader supplied with the Arduino Nano board.

First, download and install on your PC the AVRDude software. There are numerous excellent articles on the web describing how to install and configure AVRdude so we'll skip repeating the same thing here. If you have trouble finding the software, take a look at this site:

http://winavr.sourceforge.net/download.html

The tricky part to programming the Nano board with AVRDude is to invoke the proper command line instruction. We'll get to this in a moment.

First, you need a USB-ASP-Programmer, a typical one is shown in the picture below. You should also get a 10-pin socket to 6-pin plug adapter to make it easier to connect the 10-pin ribbon cable from the programmer to the 6 pin ISP header on the Nano board. The USB programmer is recognized by AVRDude as programmer type **usbasp**. See the following site for further details of this programmer.

http://www.fischl.de/usbasp/

We'll assume you are running in a Windows environment and that you placed the holtek-autodialer.hex program image file in a sub-directory such as:

c:\Users\yourname\My Documents\avr\holtek-autodialer.hex

When you invoke AVRDude be sure to specify the full path to the directory where avrdude.exe resides. We'll asume for this discussion that the avrdude files are in:

c:\opt\avr\avrdude.exe

The avrdude.conf and avrdude.rc files are also in this sub-directory.

When you are ready to program the Nano's flash with holtek-autodialer.hex executable image, plug the programmer into a USB port on the PC and plug the ribbon cable with the 10-to-6 pin adapter onto the 6-pin ISP programming header on the Nano board. Make sure you connect the adapter the right way round (adapter socket 1 connected to ISP header pin 1). The programmer will supply 5 V power to the Nano board and the power indicator LED will light up.

Open up a command line interface in a Windows terminal and go to the subdirectory c:\Users\yourname\My Documents\avr\ where the holtek-autodialer.hex file is located. Type in the following command:

"c:\opt\avr\avrdude.exe" -C"c:\opt\avr\avrdude.conf" -p m328 -c usbasp -u -U flash:w:holtek-autodialer.hex:i

AVRDude will perform a number of programming steps and print some

messages on the console screen while it uploads the holtek-autodialer.hex file to the Nano's flash memory. If the programming is successful, you'll see the program verification step completed.

As a side note to AVRDude use, a very useful companion program to have in your toolbox is the AVRdudess, a GUI for AVRDude. Check out this link if you are interested:

http://blog.zakkemble.co.uk/avrdudess-a-gui-for-avrdude/

Unplug the programmer from the Nano. Now go to the final step of powering up the autodialer project in the following section.

6.2 Powering-up the autodialer project

Once the Nano controller is programmed with the holtek-autodialer.hex executable file you can apply power to the Nano controller board via the mini USB cable and plug the other end of the cable into a host computer. Alternatively, you can connect a 5 V cell phone charger to the Nano's +5 V pin. You can also power the autodialer from an external 7 to 12 V power supply connected to the Nano's VIN pin.

When the autodialer is first powered up it needs to be programmed with a set of telephone numbers that comprise your telephone directory. On power up or controller reset, the blue LED should turn on.

Go to the PROGRAM mode and enter two 9 digit telephone numbers at index 0 and 1. Return to the RUN mode, enter 0# to request dialing out the index 0 telephone number you just entered.

You should hear the HOOK relay click on then off for around 3 s for a 9 digit number. Press 1# and the HOOK relay should again go on for around 3 s.

Time now to connect the autodialer to the telephone lines and check if it is dialing out the proper dtmf tones. Unplug your telephone set from the RJ-11 jack, insert a 'Y' plug adapter and reconnect your set to one side of the adapter. Use a standard piece of telephone extension cord to connect the autodialer to the other side of the adapter.

Lift the phone off the set and listen for a dial tone. When you hear dial tone, press 1# on the autodialer and you should hear the string of dtmf tones being dialed out. You should now hear the telephone which you called start to ring.

Try calling the telephone number stored at index 1. If you get through, then the autodialer is working. You can now program in the rest of your phone numbers into the autodialer.

7 Observations

The autodialer project can be assembled in a few evenings by anyone with average construction skills. A large part of the building time is taken up in the preparation of the case and in arranging all the component modules to fitting together without undue intereference.

Plan your component layout carefully, work slowly, and double check the wiring on the main and DAA boards.

The autodialer design is purposely kept spartan. There is no display for showing the stored telephone numbers, instead you have to keep a printed directory list of index number, telephone number and name of person.

The autodialer is by no means unique but it is designed to fulfil a single purpose and to do it with a minimum of effort. We hope this application project will inspire you to build your own version of the autodialer and that you will enjoy building and using your autodialer as much as we do.