

# (T)ANGO (R)OMEEO (O)SCAR (Y)ANKEE: A TALKING REPRESENTATION OF YOURSELF

John A. Martiney  
martiney@mil.ufl.edu

A. Antonio Arroyo  
arroyo@mil.ufl.edu

Machine Intelligence Laboratory  
Department of Electrical and Computer Engineering  
University of Florida, Gainesville, FL 32611-6200

## Abstract

*TRoY stands for “Talking Representation of Yourself.” It is a module that is able to listen to a human operator and respond back using verbal comments. Such a module can be used to give a robot the ability to talk and listen. This paper presents the development of TRoY, starting with the interface to each development board and culminating in a component that can learn new words and parse command sentences using verbal input.*

## 1. INTRODUCTION

The ability to speak in a given language is considered a sign of intelligence. However, this sign of intelligence cannot always be taken at face value. For example, a CD player can definitely “talk.,. However, it does this by playing back voices that were originally recorded. It has no control over the order of the words or which words to use at a given moment. Not many people would consider that intelligence.

TRoY is a step above systems that record and play back speech. It is able to listen to words and parse them into sentences. When it does not know a word, it has the ability to learn it. Then, it is able to immediately recognize the new word in a command sentence or use it in a spoken sentence of its own. TRoY “talks., through a text-to-speech voice synthesizer. Therefore, TRoY learns both the sound of a new word and its proper spelling.

The following is a description of how TRoY is built and adequately prepared to learn new words in the English language. It is hoped that after this description, TRoY can be considered to be one step closer towards being “intelligent.,,

## 2. THE MAIN COMPONENT INTERFACES

TRoY has three main components: an Atmel mega8 microcontroller, a RC Systems V8600A Text-to-Speech development board, and a Sensory Inc. VoiceDirect 364 word recognition development board.

### 2.1. ATMEL MEGA8

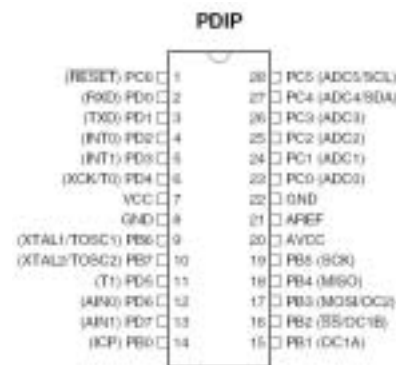


Figure 1: ATmega8 DIP

The ATmega8 takes care of controlling the entire system. It contains the strings and information that are used when interacting with TRoY. It is also responsible for parsing the input words into command sentences. The

ATmega8 is convenient for this task because it comes in a 28-pin DIP package, does not require an external oscillator, has internal 8kB Flash/512B EEPROM/1kB SRAM, has up to 23 I/O pins, and includes an internal UART system. The UART pins are connected to either the V8600A board or the RS-232 transceiver hardware (connected to a DB9 header). The I/O pins are mainly used to connect to the VoiceDirect 364 board; but, they are also used for the  $\sim$ CTS signal in the V8600A board, the TRoY mode selection jumper, and various output LEDs.

The final program uses approximately six and half of the eight kilobytes of internal flash. The flash includes both the program and the constant strings used by TRoY. Whereas, the internal EEPROM is used to store the text representations of the words that TRoY has learned or will learn. Finally, the ATmega8 is programmed using an Atmel STK500 development board.

## 2.2. RC SYSTEMS V8600A

The V8600A text-to-speech development board is used as the “vocal cords,, of the module. Developers can communicate with it through three different interfaces: microprocessor (databus), printer port, or RS-232 serial port. Figure 2, taken from the V8600A datasheet, shows these pin definitions and their respective directions. TRoY uses the RS-232 serial interface because it requires the fewest pins (TxD, RxD,  $\sim$ CTS, and GND). The TxD and RxD signals are connected to the ATmega8 internal UART pins. The  $\sim$ CTS signal is assigned to pin 4 of the ATmega8 (PD2) and is controlled using software. Finally, the SEN signal (serial transceiver enable) is left unconnected because it is not present in this development board.

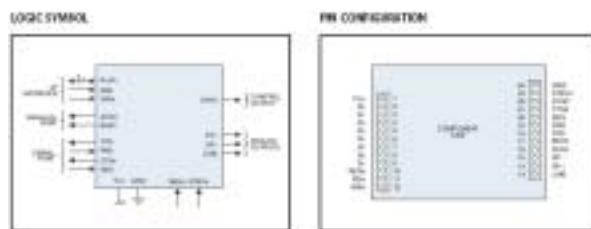


Figure 2: V8600A development board

The V8600A board receives text from the RS-232 lines as normal ASCII text. Its 2kB character buffer is filled with text until a carriage return (0x0D) or null (0x00) character is received. It is able to distinguish between commands and text by using a special command character. The default command character is CTRL-A (ASCII 0x01). Any text following the command character is considered a command. The V8600A board includes many commands for controlling different aspects of the speech synthesizer such as pitch, speed, articulation, and expression. TRoY initializes the V8600A to its default values and enables the “expression,, text-to-speech (TTS) option. The command strings are as follows:

- 0x01, '@', 0x00 → Re-initialize V8600A
- 0x01, 'E', 0x00 → Enable expression TTS mode

Now, the V8600A is ready to convert text strings into speech. Each text string is sent as normal text that is terminated by either the carriage return or the null character. The following are the welcome strings spoken by TRoY:

- "Greetings, my name is TROY, version 0.50.\n"
- "I am a Talking Representation Of Yourself.\n"

### 2.3. SENSORY INC VOICEDIRECT 364

The VoiceDirect 364 board (VDR364 in the schematics and code) is used as the “ears,, of the module. The VDR364 is able to operate in stand-alone and slave modes. In stand-alone mode, there is very little external hardware required. It is able to prompt the user for words and set certain outputs to a given level when it recognizes them. However, although the stand-alone mode is easy to use, it does not offer the flexibility of issuing individual commands. Therefore, TRoY uses the VDR364 as a slave device. In this mode, it is able to store up to 60 words (as opposed to only 15 words in stand-alone mode) and has the ability to issue individual commands to the hardware.

The interface from the ATmega8 to the VDR364 is not as easy to implement as that of the V8600A. The interface is composed of 4 wires: ~SHS (slave handshake), ~MHS (master handshake), DATA (the bi-directional 1-bit data line), and GND. The VDR364 uses full handshaking as described in the following excerpt from its datasheet:

1. When the host CPU has data to transmit to the VDR, the host CPU sets DATA to the data value, verifies that -SHS (Slave Handshake) is in the high state, then sets -MHS (Master Handshake) to the low state to request a transfer.
2. The VDR senses the low state of -MHS and reads DATA, which then sets SHS to the low state to acknowledge the DATA.
3. The host CPU senses the low state of -SHS, and sets -MHS to the high state to indicate that DATA is no longer valid, and at the same time sets DATA high (releasing it).
4. The VDR then sets -SHS to the high state to indicate that the cycle is complete. Both devices

are now ready to transfer the next data bit.

The process is completely symmetrical. Therefore, if the VDR364 wishes to send data, it uses the same handshake protocol by driving ~SHS low. Figure 3 shows the timing diagram (as shown in the VDR364 datasheet) illustrating these processes.

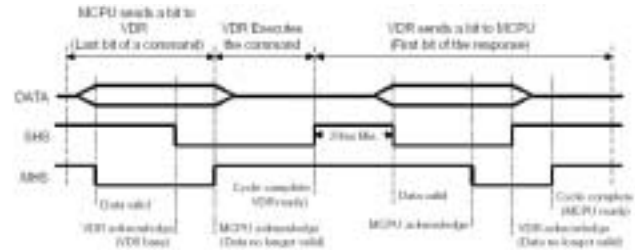


Figure 3: VoiceDirect 364 Timing Diagram

Now that the interface has been defined, there needs to be a standard for the way in which data is transferred. The VDR364 expects data to be transmitted in bytes that are arranged in message packets. Each byte is transferred with the most significant bit (MSB) first and the bytes in each packet are arranged as follows:

Byte	Value	Notes
0	0xFF	Sync Field
1	0x05	Packet length
2	0x01	First data byte
3	0x02	Second data byte
4	0x03	Third data byte
5	0x04	Fourth data byte
6	0x0F	Modulo 255 Checksum (5 + 1 + 2 + 3 + 4 = 15)

Figure 4: VoiceDirect 364 data packet format

The ATmega8 has connections to the ~MHS (PC0 pin 23), ~SHS (PC2 pin 25), and DATA (PC4 pin 27) lines of the VDR 364. This full handshaking mode and message packet format is implemented in software using those lines. The individual function implementations are explained in a later section.

### 3. COMPONENT LAYOUT AND WIRING

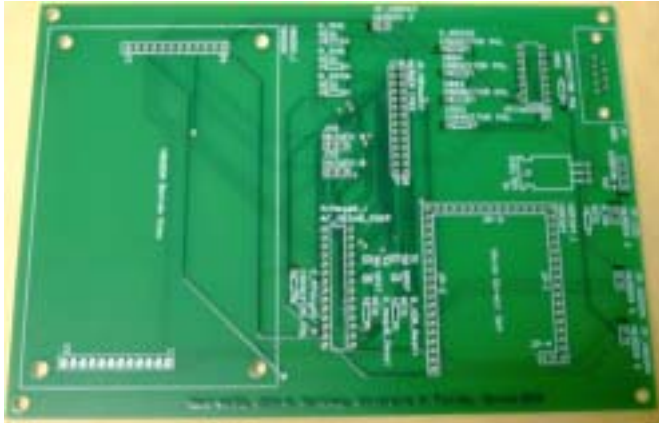


Figure 5: TRoY v0.50 PCB

The TRoY v0.50 PCB is specifically designed to connect V8600A and VoiceDirect364 development boards to an ATmega8 microcontroller. The PCB is designed using Protel 99 SE. It is a 2-layer board that measures 5in x 7in. The following section describes the various signals and connections of the PCB.

#### 3.1. LIST OF COMPONENTS BY FOOTPRINT

1. ATmega8\_1 → ATmega8 microcontroller
2. CRs1-CRS4 → RS232 charge pump capacitor (0.1 uF no polarity)
3. C\_Atmega8 → ATmega8 bypass capacitor (0.1 uF no polarity)
4. C\_RS232 → RS232 chip bypass capacitor (0.1 uF no polarity)
5. DS232A(16) → RS232 line driver
6. J4 (DB9) → DB9 female header
7. JP1 → 4-pin power header (7.5v-12v)
8. JP2 → 3-pin jumpered ATmega8 TxD connect (pin 1 faces V8600A) (1-2 DB9) (2-3 V8600A)
9. JP3 → 3-pin jumpered ATmega8 RxD connect (pin 1 faces V8600A) (1-2 DB9) (2-3 V8600A)
10. JP\_LED → LED connector (+ faces power header JP1)

11. JP\_M8Header → female receptacle to access ATmega8 signals
12. JP\_V86PSK → V8600A Speaker connector (+ faces JP\_LED)
13. JP\_VDRMIC → VoiceDirect 364 microphone connector (+ faces V8600A)
14. JP\_VDRSPK → VoiceDirect 364 Speaker connector (+ faces JP\_LED)
15. R\_DATA → VoiceDirect 364 Data pull-up resistor (100 kOhm)
16. R\_mega8\_Reset → ATmega8 pull-up resistor (2.7 kOhm)
17. R\_MHS → VoiceDirect 364 ~MHS pull-up resistor (100 kOhm)
18. R\_ON → power LED current limiting resistor (330 Ohm)
19. R\_SHS → VoiceDirect 364 ~SHS pull-up resistor (100 kOhm)
20. R\_VDR\_Reset → VoiceDirect 364 pull-up resistor (2.7 kOhm)
21. S1 → switch to reset VoiceDirect 364
22. S2 → switch to reset ATmega8
23. U1 → 7805 Voltage Regulator
24. VDR364\_1 → female receptacle for VoiceDirect 364 development board
25. V8600A\_1 → female receptacle for V8600A development board

#### 3.2. THE ATMEGA8 MICROCONTROLLER

ATmega8_1				
MS_P008	P00 (-RESET)	(ADC56CL) PC3	28	MS_P05
MS_P010	P00 (E2D)	(ADC48DA) PC4	27	VDR_DATA
MS_P012	P01 (TED)	(ADC3) PC3	26	MS_P03
-CT2_4	P02 (B7D)	(ADC2) PC2	25	VDR_BIE
MS_P005	P03 (B7E)	(ADC1) PC1	24	MS_P01
MS_P046	P04 (205/TE)	(ADC0) PC0	23	VDR_MHS
VS+	VCC	GND	22	GND
GND_8	GND	AREF	21	MS_AREF
MS_P069	P06 (XTAL1/TOSC1)	VCC	20	VS+
MS_P030	P07 (XTAL2/TOSC2)	(ICD) PE5	19	MS_P05
MS_P031	P03 (T1)	(M00) PE4	18	MS_P04
MS_P042	P04 (AB0)	(M02) PE3	17	MS_P03
MS_P003	P07 (AB1)	(-D00) PE2	16	MS_P02
MS_P004	P00 (ICP)	(OC1A) PE1	15	MS_P01

Figure 6: ATmega8 wiring diagram



Figure 7: ATmega8 Header

The ATmega8 wiring diagram includes the following signals:

- mega8RxD → The mega8 UART Receive signal
- mega8TxD → The mega8 UART Transmit signal
- ~CTS → The ~CTS signal used for V8600A
- VDR\_DATA → VDR 364 Data
- VDR\_SHS → VDR 364 Slave handshake
- VDR\_MHS → VDR 364 Master handshake
- M8\_signals → extra labels to connect to the mega8 header

### 3.3. THE V8600A TEXT-TO-SPEECH BOARD

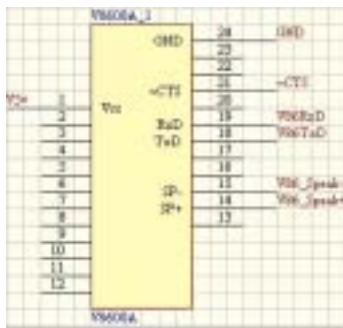


Figure 8: v8600a Text-to-Speech board wiring diagram

The V8600A wiring diagram includes the following signals:

- ~CTS → ~CTS signal used in RS-232 communication

- V86RxD → V8600A RS-232 Receive signal
- V86TxD → V8600A RS-232 Transmit signal
- V86\_Speak- → Negative speaker terminal
- V86\_Speak+ → Positive speaker terminal

### 3.4. THE VOICEDIRECT 364 SPEECH RECOGNITION BOARD

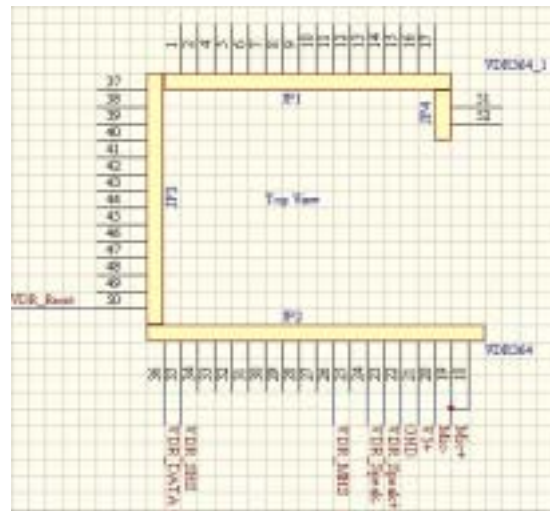


Figure 9: VoiceDirect 364 wiring diagram

The VDR 364 wiring diagram includes the following signals:

- VDR\_Reset → VoiceDirect 364 reset signal
- VDR\_DATA → VDR 364 Data
- VDR\_SHS → VDR 364 Slave handshake
- VDR\_MHS → VDR 364 Master handshake
- VDR\_Speak- → Negative speaker terminal
- VDR\_Speak+ → Positive speaker terminal
- Mic- → Negative microphone terminal
- Mic+ → Positive microphone terminal



- v8600.c → v8600a TTS functions
- vdr364.c → VoiceDirect 364 functions
- vdr364tst.c → special VoiceDirect 364 Monitor functions

#### 4.1. M8UART.C

The m8uart.c functions can be added to a project by using the m8uart.h file. The following is a description of each function.

- void USART\_Init(unsigned int baud) → Enable the internal UART and initialize to the input baud rate
- void USART\_Transmit(unsigned char data) → Send a single character
- unsigned char USART\_Receive(void) → Receive a single character
- void USART\_string(unsigned char \*outString) → Send a string
- void USART\_getstring(unsigned char \*inString) → Get a string

#### 4.2. TROY.C

The TRoY.c functions can be added to a project by using the TRoY.h file. The following is a description of each function.

- void TRoYinit(void) → Initialize LED and boot selection pins
- void clearWords(void) → Clear text words stored in EEPROM
- void getWord(unsigned char \*wordptr, unsigned char wordnum) → Get the text word at index wordnum from EEPROM
- void storeWord(unsigned char \*wordptr, unsigned char wordnum) → Store a text word into index wordnum in EEPROM
- void showWords(void) → Display currently stored words

The header file also includes various ‘#define’ statements that can be used to easily change the location of the LED and boot selection pins on the ATmega8.

#### 4.3. TROYXX.C

The most current version of the TRoY software is TRoY22.c. This file includes all of the text strings shown in prompts and spoken by TRoY. The main routine uses the functions described in all of the other C files. All of the text strings are stored in flash memory.

#### 4.4. V8600.C

The v8600.c functions can be added to a project by using the v8600.h file. The following is a description of each function.

- void USART\_stringCTS(unsigned char \*outString) → Send out a string using the UART with ~CTS handshaking to the TTS board
- void v8600init(void) → Initialize the v8600a board (the ~CTS pin on the ATmega8)

The header file also includes various ‘#define’ statements that can be used to easily change the location of the ~CTS pin on the ATmega8.

#### 4.5. VDR364.C

The vdr364.c functions can be added to a project by using the vdr364.h file. The following is a description of each function.

- void VDRinit(void) → initialize the VDR364 DATA, ~MHS, and ~SHS pins
- void VDRsend(u08 VDRbyte) → Send a single byte of data to VDR364
- u08 VDRrecv(void) → Receive a single byte of data from VDR
- void VDRcommand(VDRpack VDRcomm) → Send a command packet to the VDR364
- VDRpack VDRresponse(void) → Receive a response packet from VDR364

- void showVDRRes(VDRpack) → Use UART to print out the response packet
- void setAttr(unsigned char) → VDR command; set current attribute byte
- void setMask(unsigned char) → VDR command; set current search mask
- unsigned char learnWord(unsigned char) → VDR command; learn a new word and return the response code
- unsigned char listenWord(void) → VDR command; wait for a single word and return the response code
- unsigned char getPtr(void) → VDR command; get the current word pointer
- unsigned char getAttr(void) → VDR command; get the attribute byte of the current pointer
- void clearVDR(void) → VDR command; erase VoiceDirect 364
- unsigned char checkYesNo(void) → waits for a yes/no answer and returns the index of either 'yes' or 'no'

In addition to defining the VDRpack structure, the header file also includes various '#define' statements that can be used to easily change the location of the DATA, ~MHS, and ~SHS pins on the ATmega8.

#### 4.6. VDR364TST.C

The vdr364tst.c functions can be added to a project by using the vdr364tst.h file. The following is a description of each function.

- VDRpack typeVDR(void) → Allows the user to type commands (in hex) for the VDR364 as described in the datasheet
- VDRpack buildVDR(unsigned char \*, unsigned char) → Used by typeVDR; creates the command VDRpack from the typed string
- unsigned char ASCtoHEX(unsigned char) → Converts an ASCII character into a hex nibble

- unsigned char HEXHtoASC(unsigned char) → Converts high nibble into ASCII
- unsigned char HEXLtoASC(unsigned char) → Converts low nibble into ASCII

## 5. TROY AT WORK

TROy has 2 operating modes. These modes are selected by connecting PB1 (pin 15) of the ATmega8 to either GND (Debug) or +5V (Interaction). The JP2 and JP3 jumpers on the PCB must also be changed depending on the operating mode. They connect the ATmega8 RxD and TxD pins to either the DB9 header (Debug) or the V8600A text-to-speech board (Interaction). Therefore, they should be placed so that they either short the two pins away from the DB9 header (Debug) or the two pins away from the V8600A board (Interaction). The speaker should be connected to the JP\_VDRSPK header (Debug) or the JP\_V86SPK header (Interaction). Finally, the microphone is always connected to the JP\_VDRMIC header.

### 5.1. MODE 1: DEBUG AND LEARN MODE

In this mode, the user is able to manage and test the vocabulary currently known by TROy through a serial port. The board is configured to work at 9600 baud with eight data bits, no parity, one stop bit, and no flow control.

Resetting TROy causes the ATmega8 and the VDR364 to reset and a beep should be heard from the speaker. Then, the following menu is shown in the terminal window.

```
Welcome to the TROy Lexicon
Input System
  1 - Menu-driven system
  2 - TROy VDR364 Monitor

>
```

### 5.1.1. MENU-DRIVEN SYSTEM

The first selection allows the user to manage the internal vocabulary by using the following menu items.

```

Menu-driven dictionary input
 0 - Completely erase
dictionary
 1 - Add an adjective
 2 - Add an adverb
 3 - Add an article
 4 - Add a noun
 5 - Add a preposition
 6 - Add a pronoun
 7 - Add a verb
l - (L)isten w/ LEDs
m - Return to (M)ain
v - Add a (V)ocabulary
>

```

Erasing the dictionary will remove the text words stored in EEPROM and completely erase the VDR364 spoken words. TRoY will then ask the user to store its four main command words: start, stop, yes, and no. The standard way for TRoY to learn is to ask for the text version of the word and then ask the user to say it into its speaker. The text version is stored in EEPROM and the spoken version is stored in the VDR364.

After erasing the dictionary, TRoY is ready to learn its alphabet. The alphabet must **always** be taught immediately after storing its four command words. TRoY uses the alphabet in its “Interaction Mode,, to learn the spelling of new words. The user must press ‘v’ to teach TRoY all 26 letters using the standard military Alpha-Bravo phonetic alphabet as follows:

Letter	Phonetic Word
A	Alpha
B	Bravo
C	Charlie
D	Delta
E	Echo
F	Foxtrot

G	Golf
H	Hotel
I	India
J	Juliette
K	Kilo
L	Lima
M	Mike
N	November
O	Oscar
P	Papa
Q	Quebec
R	Romeo
S	Sierra
T	Tango
U	Uniform
V	Victor
W	Whiskey
X	X-ray
Y	Yankee
Z	Zulu

Figure 16: The Alpha-Bravo Phonetic Alphabet

Once TRoY has learned its alphabet, it is ready to be used in the “Interaction Mode,,. However, at this point TRoY is able to learn new words that can be spelled using the keyboard instead of the phonetic alphabet. To add a word, the user must select the type of word (adjective, adverb, etc), spell the word at the prompt, and say the word into TRoY’s speaker. The process is illustrated below.

```

**The user selects option four in the menu to
add a noun**
Please type the word: gator <enter>
**The VDR364 prompts user for input word by
saying: “Say a word,**
**The user says the word “gator,**
**The VDR364 asks user to repeat the word by
saying: “Repeat,**
**The user repeats the word “gator,**
**The VDR364 indicates that training has ended
by saying: “Training complete,**
VDR Response: 00

```

A VDR response of 00 indicates that the word has been added successfully. Any other response indicates an error (usually 07). TRoY reports back the standard response codes used in the VDR364 board. The user can reference the VoiceDirect 364 datasheet for a complete list of error codes.

Once TRoY has learned all of the words, it allows the user to test the stored words by using the “(L)isten w/ LEDs,, menu option. TRoY will listen to the user and will indicate the type of word that it heard by outputting the lower four bits of its corresponding binary pattern on the LEDs. Saying the “stop,, word will return the user to the previous menu. The binary patterns (VDR364 masks) are defined in the “TRoY.h,, header file and are as follows:

Type of Word	Binary pattern (VDR364 Mask)
Adjective	00000001
Adverb	00000010
Article	00000011
Noun	00000100
Preposition	00000101
Pronoun	00000110
Verb	00000111
Start/Stop/Yes/No	01010101
Alphabet	10101010

Figure 17: VDR364 Word Masks

When TRoY learns a spoken word, it tags it with its appropriate mask and then saves in the VDR364. These masks are mainly used for parsing sentences. However, they also allow TRoY to allocate its 30 word vocabulary any way it needs it. For example, it can learn 10 nouns, 10 articles, and 10 verbs or 20 nouns, 5 adverbs, 3 verbs, and 2 articles.

The word learning software component is independent of the sentence parsing software component. Therefore, TRoY is able to learn certain types of words that it may not be able to

parse in sentences. Currently, TRoY can only parse command sentences. These command sentences use only articles, nouns, and verbs. Therefore, adjectives, adverbs, prepositions, and pronouns are ignored when parsing.

### 5.1.2. TROY VDR364 MONITOR

The second main menu selection allows the user to send hex commands to the VDR364 as described in the VoiceDirect 364 datasheet. It does not require that the user enter the number of bytes to send or the checksum. For example, command 01h reports back the version number information. The user can execute the command as follows:

```
Welcome to the TRoY VoiceDirect 364
Monitor Program
Please type in the VoiceDirect 364
command string

> 01
Command String sent: 01

VDR Response: 00 56 44 49 00 02

>
```

This mode allows the user to have absolute control of the VDR364 hardware. Therefore, it is possible to disrupt TRoY’s functionality if the commands are not executed with care. TRoY can always be restored to a functional state by resetting the hardware, erasing its vocabulary, and re-training the command and alphabet words.

### 5.2. MODE 2: INTERACTION MODE

In this mode, TRoY is able to parse command sentences and learn new words. The command sentences are always in the form: <verb> <article> <noun>. For example:

- “Get the pencil,,
- “Locate the chair,,
- “Bring a battery,,

Upon entering this mode, TRoY reports its name and version information. Then, it waits for the user to say the “start,, command. This allows TRoY to listen for words and parse them into sentences. Each time it correctly identifies a word, TRoY displays the binary pattern of the type of word it heard on its LEDs. If the user correctly issues a command sentence, TRoY responds back by indicating that it will perform the desired action. The following is a sample script:

```
TROY: Greetings, my name is
TROY, version 0.50.
TROY: I am a Talking
Representation Of Yourself.
USER: start
TROY: Hello human. I will
listen to you now.
USER: locate
**The LEDs display the binary
pattern for a verb - 0111**
USER: a
**The LEDs display the binary
pattern for an article - 0011**
USER: battery
**The LEDs display the binary
pattern for a noun - 0100**
TROY: I will locate a battery.
```

TRoY continues to listen until the user issues the “stop,, command. It then acknowledges the command and ignores spoken words until the “start,, command is issued again. If TRoY is not able to recognize a spoken word, it asks if it should learn the given word. The following script illustrates how TRoY can learn a new noun (robot).

```
TROY: I do not know that word.
TROY: Should I learn it?
USER: Yes
TROY: Is it an adjective?
USER: No
TROY: Is it an adverb?
USER: No
TROY: Is it an article?
USER: No
TROY: Is it a noun?
USER: Yes
TROY: My LEDs should show the
pattern for a noun.
```

```
**The LEDs display the binary
pattern for a noun - 0100**
TROY: Please, spell out the new
word
TROY: using the Alpha-Bravo
vocabulary.
USER: Romeo
TROY: I heard, Romeo.
USER: Oscar
TROY: I heard, Oscar.
USER: Bravo
TROY: I heard, Bravo.
USER: Oscar
TROY: I heard, Oscar.
USER: Tango
TROY: I heard, Tango.
USER: Stop
TROY: You will now have to
connect a speaker to my speech
TROY: recognition hardware, in
order to hear the prompts
**The user connects the speaker
to the JP_VDRSPK header**
**The VDR364 prompts the user
for a new word - "Say a Word"**
USER: Robot
**The VDR364 asks the user to
repeat the new word - "Repeat"**
USER: Robot
**The VDR364 acknowledges the
new word - "Training Complete"**
**The user connects the speaker
to the JP_V86SPK header**
TROY: Processing, Processing,
Processing
TROY: I have added a new word
to my vocabulary.
TROY: It is robot.
```

While spelling the word, the user can issue the “No,, command to erase the last spoken letter. If there are no letters to erase, TRoY will say: “There are no more letters in the buffer., Also, the process of switching the speaker between the JP\_V86SPK and JP\_VDRSPK headers is something that can be handled with a multiplexer and the addition of software to switch between the two at the right time.

It should be noted that TRoY’s learning process involves three steps: identify the type of word, learn the spelling of the word, and learn the spoken word. After completing these three

steps, the user can issue the command: “Locate a robot.,, TRoY should respond to the command by saying: “I will locate a robot.,,

## 6. CONCLUSION

TRoY is a module that can be used to allow a robot to talk, listen, and learn new words. It is able to parse command sentences and generate responses based on those sentences. Currently, the response is a spoken acknowledgement. However, this response can be changed to any number of possibilities allowed by a microcontroller. In its current state, TRoY is both a finished product and an unfinished prototype. It is a finished product because it performs its function. It is able to successfully talk, listen, and learn in a very limited way. This “very limited way,, is what qualifies it as an unfinished prototype. It, like any other robot component, can be modified and improved.

## 7. REFERENCES

- [1] Atmel Corporation. ATmega8(L) Preliminary Complete. 24 March 2003. <[http://www.atmel.com/dyn/resources/prod\\_documents/doc2486.pdf](http://www.atmel.com/dyn/resources/prod_documents/doc2486.pdf)>.
- [2] Atmel Corporation. AVR Studio v3.5x. 24 March 2003. <[http://www.atmel.com/dyn/resources/prod\\_documents/astudio3.exe](http://www.atmel.com/dyn/resources/prod_documents/astudio3.exe)>.
- [3] Atmel Corporation. STK500 User Guide. 24 March 2003. [http://www.atmel.com/dyn/resources/prod\\_documents/DOC1925.PDF](http://www.atmel.com/dyn/resources/prod_documents/DOC1925.PDF)>.
- [4] AVR Freaks .NET. Win32 build of avr-gcc 3.2. 24 March 2003. <[http://www.avrfreaks.net/filedownload.php?url=/AVRGCC/Download/2002-06-25\\_FREAKS.exe](http://www.avrfreaks.net/filedownload.php?url=/AVRGCC/Download/2002-06-25_FREAKS.exe)>.
- [5] NASA/Glenn Research Center. Phonetic Alphabet. 24 March 2003. <<http://www.grc.nasa.gov/WWW/MAEL/ag/phonetic.htm>>.
- [6] RC Systems, Inc. V8600A Voice Synthesizer Module. 24 March 2003. <<http://www.rcsys.com/Downloads/v8600a.pdf>>.
- [7] Sensory, Inc. Voice Direct 364 Manual. 24 March 2003. <<http://www.sensoryinc.com/html/support/docs/80-0179-F.pdf>>.