Arduino Electronics Blueprints

Make common electronic devices interact with an Arduino board to build amazing out-of-the-box projects

Don Wilcher
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 3 'A Talking Logic Probe'
- A synopsis of the book’s content
- More information on Arduino Electronics Blueprints
Don Wilcher is a passionate educator of electronics and robotics technology and an electrical engineer with 26 years of experience. He has authored several books on Lego Robotics and Arduinos. His latest book published by Maker Media, titled Make: Basic Arduino Projects, has been approved by the Alabama State Department of Education to be on their reading list. He's also a Certified Electronics Technician (CETa) and Exam Administrator (CA) with ETA International as well as State Certified Teacher for Career Technical Education (CTE) as a Level 2 Specialist in electronics technology. He has worked on industrial robotic systems, automotive electronic modules/systems, and embedded wireless controls for small consumer appliances. While at the Chrysler Corporation, Don developed a weekend enrichment pre-engineering program for inner city kids. He’s a contributing writer and webinar lecturer for Design News Magazine. He is also an electronics and robotics technologist who has developed 21st century educational products and training materials for Makers, hardware start-up entrepreneurs, and educators.
Preface

You have purchased your first Arduino, and now you're wondering what project to build with it. There are hundreds of websites with an assortment of electronic gadgets and devices to build, but the search and choosing the first project can be overwhelming. Besides building awesome Arduino gadgets, some of the website projects leave out how the electronics and code work with a programmable prototyping platform. Also, the projects found on the Web don't provide additional challenges to test your new Maker skills as well.

The Arduino Electronics Blueprints book was written to address the concerns mentioned in a user friendly and educational format. Every chapter in the book starts off with either a historical reference to electronic discoveries or a brief discussion of present technologies used in contemporary consumer, entertainment, or industrial products. The book was designed to show how to build awesome electronic devices using parts found in laboratory bins or junk boxes. Also, new prototyping materials such as littleBits electronic modules and Elenco SNAP circuit kits are introduced to readers as well. The new and exciting prototyping materials presented allow us to rapidly build the target Arduino device discussed in some of the book’s chapters. To aid readers in building the fun Arduino projects, a Parts list of electronic components is included in each chapter of the book. Detailed circuit schematic and wiring diagrams and Arduino code are provided in each chapter. Also, basic circuit theory and Arduino code explanations are provided in each project chapter as well. To conclude the chapter, a DIY challenge is presented, so readers may explore additional prototyping topics in new product designs of their own. I enjoyed designing, building, and testing each chapter’s project and hope readers of the Arduino Electronics Blueprints book will find the projects to be fun and entertaining as well.
What this book covers

Chapter 1, *A Sound Effects Machine*, will teach the reader how to build an Arduino sound effects machine using an SD module, digital logic switches, a transistor speaker amplifier, and .wav files. Also, the reader will learn how to add a random function in order as to play different sounds automatically without using digital logic switches.

Chapter 2, *Programmable DC Motor Controller with an LCD*, shows the reader how to build an Arduino programmable controller to operate small DC motors. Also, to aid in operating the programmable controller, the reader will learn how to add a Liquid Crystal Display (LCD) to the electronic device as well.

Chapter 3, *A Talking Logic Probe*, explains a talking electronic instrument that the reader can build to test microcontroller and digital circuits. Also, the reader will learn how to wire an Arduino to an EMIC 2 (text-to-speech) module and program it using special character codes.

Chapter 4, *Human Machine Interface*, A Human Machine Interface (HMI) used in industrial controls to operate electromechanical devices, such as motors, will be discussed in this chapter. The reader will also learn how to build a HMI using an Arduino, a virtual server, and JavaScript to control a small DC motor.

Chapter 5, *IR Remote Control Tester*, allows the reader to learn how to build a testing device to check the operation of any IR remote control. Also, the reader will learn about IR detectors and digital codes using this electronic tester.

Chapter 6, *A Simple Chat Device with LCD*, will teach the reader how to send text messages to an Arduino using a Bluetooth Low Energy (BLE) device and an Android smartphone. Also, the RedBearLabs BLE Arduino shield used to send and receive text messages will be introduced to the reader in this chapter.

Chapter 7, *Bluetooth Low Energy Controller*, will show the reader how to send BLE control signals to an Arduino using the RedBearLabs BLE shield and an Android smartphone to control a DC motor. Also, a seven segment LED display’s electrical operation will be discussed along with making letter characters using the BLE Controller.

Chapter 8, *Capacitive Touch Sensing*, explores a simple DC motor controller using an Arduino and a 555 timer IC-based capacitive touch sensor. The reader will learn the basic operation of the 555 timer by building an Arduino-enabled touch sensing controller.

Chapter 9, *Arduino-SNAP Circuit AM Radio*, introduces the reader the Elenco SNAP circuit kit by building an AM radio. Also, the reader will learn how to operate the AM radio using the RedBearLab BLE shield and an Android smartphone.
Chapter 10, Arduino Scrolling Marquee, discusses organic light-emitting diode (OLED) technology by building an Arduino-based scrolling marquee. Also, the reader will learn to use any ordinary IR handheld remote to control the scrolling effect of the OLED marquee.
A Talking Logic Probe

Digital circuits are electronic devices that show either high or low operating states. Other ways of describing digital circuit operating states include true or false and binary 1 or 0. A Truth Table (TT), as discussed in the previous chapter, is a graphical tool used to test the electrical operation of digital circuits by showing input data and observing their output states. The TT aids the makers building digital circuits by allowing them to inject test data into the device's input pins and observing the output states using a logic probe.

A logic probe is an electrical tester that displays input and output signals of a digital circuit using visual indicators. LEDs are typical visual indicators used to show the binary data of a digital circuit. Typical colors used to represent high and low digital data are red and green, respectively. Some variations of logic probe visual indicator may consist of a seven segment display capable of displaying the letters H or L for high or low binary data, respectively. A typical logic probe to test digital circuits is shown here:
In this chapter, we will learn how to build a talking logic probe for testing digital circuits and devices. Also, to make the logic probe unique from the typical off-the-shelf testers, voice synthesis will be used to speak the operating conditions of digital circuits. An EMIC 2 text-to-speech (TTS) module will be used to generate the voice synthesis for the logic probe. Specific EMIC 2 TTS module operation modes will be explored in this chapter using different Arduino sketches. This chapter will also provide instructions for building the logic probe.

**Parts list**
The following is the list of parts required for building a talking logic probe:

- Arduino Uno (one unit)
- (1) EMIC 2 text-to-speech module (one unit)
- 8 ohm speaker (one unit)
- 1 kilo ohm resistor (black, brown, red, and gold) (one unit)
- Momentary pushbutton electrical switch (one unit)
- Breadboard
- Wires

**A talking logic probe block diagram**
The concept of a talking logic probe can be thought of as a typical testing device with a speaker. The LEDs for binary high and low status can be removed from the logic probe because of the voice synthesis feature of the digital tester. To test different digital IC families with a logic probe is very important in electronics because today’s products use them extensively. Therefore, a slide switch for selecting between Transistor-Transistor Logic (TTL) and Complementary Metal-Oxide-Semiconductor (CMOS) type digital ICs is provided in the talking logic probe.
CMOS devices use FETs (Field Effect Transistors) instead of Bipolar Junction transistors (BJTs) used by TTL ICs. The CMOS technology is being used in all the semiconductor devices because of lower power dissipation as compared to the TTL technologies. Also, some CMOS ICs can operate on as low as 1.8 V DC to 2.0V DC whereas TTL devices supply a voltage range between 3.75V DC to 5.75V DC.

A concept diagram of a talking logic probe is shown here:

The talking logic probe block diagram is an engineering development tool used to convey a complete product design using simple graphics. The block diagram also makes it easier to plan the breadboard for prototyping and testing of the talking logic probe in a maker’s workshop or laboratory bench. A final observation of the talking logic probe block diagram is that the basic computer convention of inputs is on the left-hand side, the processor is located in the middle, and the outputs are placed on the right-hand side of the design layout. As shown, the **Digital Circuit under Test (DCuT)** is on the left-hand side, the Arduino is located in the middle, and the EMIC 2 TTS module with the 8 ohm speaker is shown on the right-hand side of the block diagram. The DCuT will provide binary high or low signals, based on the digital circuits inputs being triggered.
This left to right design method makes it easier to build the talking logic probe and troubleshoot the errors during the testing phase of the project.

A talking logic probe – testing the EMIC 2 TTS module

The talking logic probe block diagram is simple in design as compared to the block diagram we used in Chapter 2, Programmable DC Motor Controller with LCD. As discussed in Chapter 1, A Sound Effects Machine, there are a variety of ways to build (prototype) electronic devices, such as on a Printed Circuit Board (PCB) or an experimenter/prototype board. The prototyping tool used to build and test the device’s EMIC 2 TTS module is a solderless breadboard shown next. The placement of electronic parts, as shown in the following diagram, is not restricted to the solderless breadboard layout, but is used as a guideline:
Another method of parts placement onto the solderless breadboard is to use the block diagram shown previously. This method of parts arrangement, illustrated in the block diagram, allows ease in testing each subcircuit separately. For example, the EMIC 2 TTS module can be tested by using a sample code given next. With the sketch uploaded to the Arduino, the musical song *Daisy Bell* can be heard through the speaker. After the song has played once, pressing the reset button repeats the musical lyrics:

```c
/*
 Emic 2 Text-to-Speech Module: Basic Demonstration

 Author: Joe Grand [www.grandideastudio.com]
 Contact: support@parallax.com
```
Program Description:

This program provides a simple demonstration of the Emic 2 Text-to-Speech Module. Please refer to the product manual for full details of system functionality and capabilities.

Revisions:

1.0 (February 13, 2012): Initial release
1.1 (April 29, 2014): Changed rxPin/txPin to use pins 10/11, respectively, for widest support across the Arduino family (http://arduino.cc/en/Reference/SoftwareSerial)

#include <SoftwareSerial.h>

#define rxPin 10  // Serial input (connects to Emic 2's SOUT pin)
#define txPin 11  // Serial output (connects to Emic 2's SIN pin)
#define ledPin 13  // Most Arduino boards have an on-board LED on this pin

// set up a new serial port
SoftwareSerial emicSerial = SoftwareSerial(rxPin, txPin);

void setup()  // Set up code called once on start-up
{
    // define pin modes
    pinMode(ledPin, OUTPUT);
    pinMode(rxPin, INPUT);
    pinMode(txPin, OUTPUT);

    // set the data rate for the SoftwareSerial port
    emicSerial.begin(9600);

    digitalWrite(ledPin, LOW);  // turn LED off
/*
When the Emic 2 powers on, it takes about 3 seconds for it to
successfully initialize. It then sends a ":" character to indicate it's
ready to accept commands. If the Emic 2 is already initialized, a CR will also
cause it to send a ":"*

emicSerial.print('\n');  // Send a CR in case the
system is already up
while (emicSerial.read() != ':');  // When the Emic 2 has
initialized and is ready, it will send a single ":" character,
so wait here until we receive it
delay(10);  // Short delay
emicSerial.flush();  // Flush the receive buffer
}

void loop()  // Main code, to run repeatedly
{
  // Speak some text
  emicSerial.print('S');
  emicSerial.print("Hello. My name is the Emic 2 Text-to-Speech
 module. I would like to sing you a song.");  // Send the desired
  string to convert to speech
  emicSerial.println('\n');
  digitalWrite(ledPin, HIGH);  // Turn on LED while Emic is
  outputting audio
  while (emicSerial.read() != ':');  // Wait here until the Emic
  2 responds with a ":" indicating it's ready to accept the next
  command
  digitalWrite(ledPin, LOW);
  delay(500);  // 1/2 second delay

  // Sing a song
  emicSerial.println("D1\n");
  digitalWrite(ledPin, HIGH);  // Turn on LED while Emic is
  outputting audio
  while (emicSerial.read() != ':');  // Wait here until the Emic
  2 responds with a ":" indicating it's ready to accept the next
  command
  digitalWrite(ledPin, LOW);
A Talking Logic Probe

while(1)      // Demonstration complete!
{
    delay(500);
    digitalWrite(ledPin, HIGH);
    delay(500);
    digitalWrite(ledPin, LOW);
}

In addition to the test sketch, the circuit schematic diagram for the solderless breadboard wiring diagram is shown for reference:
EMIC 2 TTS module basics

The heart of the talking logic probe is the voice synthesis engine for speaking the binary high and low signals produced by the digital circuits. The EMIC 2 TTS module, as shown next, is a versatile, multi-language voice synthesizer capable of taking digital text and converting it into audible speech. The EMIC 2 TTS module uses the DECTalk text for text-to-speech synthesizer engine. As shown in the circuit schematic and breadboard wiring diagrams, the EMIC 2 TTS module is quite easy to wire to the Arduino. This ease of adding an audible voice to electrical testers can provide an eyes-free approach to troubleshooting electronic circuits. Instead of looking at a visual display for measurement data, the EMIC 2 TTS can provide speech output for the electrical measuring device. This section of the chapter will discuss the basic features and electrical connections of the EMIC 2 TTS module.

The EMIC 2 TTS module is manufactured by Parallax Inc and designed by Joe Grand of Grand Design Studio (www.grandideastudio.com).
EMIC 2 TTS module's key features

The EMIC 2 TTS module provides a variety of features that make it appealing to embedded applications, such as robotics, healthcare, automotive, and industrial products. Some key features of the EMIC 2 TTS module include high quality speech synthesis, pre-defined voice styles, an on-board audio amplifier, and a single row header for electrical wiring. The EMIC 2 only requires a +5V DC, 30 mA DC power supply. The operating temperature range of the EMIC 2 TTS module is \(-20^\circ\) C to \(70^\circ\) C (\(-4^\circ\) F to \(150^\circ\) F), making it suitable for typical environments. Dimensions of the EMIC 2 TTS module are 1.25" w x 1.5" l x 0.37" h, which allows it to fit easily inside a small plastic hobby box. The following are the picture views of the EMIC 2 TTS module:

Another thing to note about the EMIC 2 is the small audio jack soldered onto the PCB. This standard audio jack (size 1/8" or 3.5 mm) allows the device to be connected to most audio Hi-Fi systems or headphones, if desired. An example design application for this audio jack is the creation of a portable foreign language training tool. Connection to headphones instead could provide individual training to the foreign language learner without disturbing people in the room. The EMIC 2 TTS module has two speech synthesis languages available for English and Spanish. Another training aid application suitable for the audio jack feature is a physical digital logic simulator with audible I/O feedback. The audio jack could also be used in this project instead of the speaker.

If both 3.5 mm jack and external 8 ohm speaker are used at the same time, audio quality might be affected.
Electrical connections

As seen previously, there are only six electrical connections or pins required for wiring the EMIC 2 TTS module to an Arduino. The electrical connections consist of GND, 5 V, SOUT, SIN, SP-, and SP+. The following descriptions explain the operation of these six electrical connections:

- **Pin 1: GND**: This is a device ground pin. Connect the Arduino's supply voltage ground (GND) to this pin.
- **Pin 2: 5 V**: This is a device power pin. Connect the Arduino's positive voltage (5 V) to this pin.
- **Pin 3: SOUT**: This pin is a device serial output to the Arduino. It is a 5 V TTL level interface, 9,600 bps*, 8 data bits, no parity, 1 stop bit, non-inverted digital signal.
- **Pin 4: SIN**: This pin is a device serial input from the Arduino. It is a 3.3 V to 5 V TTL level interface, 9,600 bps, 8 data bits, no parity, 1 stop bit, non-inverted digital signal.
- **Pin 5: SP-**: Device differential audio amplifier output, bridge-tied load configuration, negative side. Connect to 8 ohm speaker directly.
- **Pin 6: SP+**: Device differential audio amplifier output, bridge-tied load configuration, positive side. Connect to 8 ohm speaker directly.

For a typical serial asynchronous communications rate, the value is 9,600 bps (bits per second).

The following circuit schematic diagram illustrates this wiring connection scheme:
Let's build it!
Now is the time for building our own talking logic probe. Follow these steps:

1. Wire the talking logic probe circuit on a solderless breadboard, as shown next. For reference, the circuit schematic diagram has been provided here:

2. Upload the talking logic probe code to the Arduino using the sketch given after the schematic diagram.

3. The phrase *Signal is Low* will be heard through the talking logic probe's speaker.

4. Touch the logic probe's green test lead onto the EMIC 2 TTS module's +5 V pin.
5. The phrase *Signal is High* will be heard through the talking logic probe's speaker.

The mystery part represents the DCuT.

The code required to make the talking logic probe operational is:

```c
/*
 * Talking Logic Probe
 *
 * Author: Don Wilcher 2014 14 11
 *
 * Program Description:
 *
 * This program allows tracing digital circuit signals using the Emic 2 Text-to-Speech
 */
```
A Talking Logic Probe

Module. Detecting a high or low signal will allow the Emic 2 TTS module to speak one of the voltage levels present.

*/

// include the SoftwareSerial library so we can use it to talk to the Emic 2 module
#include <SoftwareSerial.h>
#define rxPin   10  // Serial input (connects to Emic 2's SOUT pin)
#define txPin   11  // Serial output (connects to Emic 2's SIN pin)
#define probein 5  // Digital probe input
int probe=0; // probein variable status

// set up a new serial port
SoftwareSerial emicSerial = SoftwareSerial(rxPin, txPin);

void setup() // Set up code called once on start-up
{
   // define pin modes
   pinMode(rxPin, INPUT);
   pinMode(txPin, OUTPUT);
   pinMode(probein, INPUT);

   // set the data rate for the SoftwareSerial port
   emicSerial.begin(9600);

   emicSerial.print('
'); // Send a CR in case the system is already up
   while (emicSerial.read() != ':'); // When the Emic 2 has initialized and is ready, it will send a single ':' character, so wait here until we receive it
   delay(10); // Short delay
   emicSerial.flush(); // Flush the receive buffer
}

void loop() // Main code, to run repeatedly
{
   // Speak some text
   emicSerial.print('S');
   probe=digitalRead(probein);// read probein status
   if(probe == HIGH) { // Emic 2 will speak HIGH message if status value is true
      emicSerial.print("SIGNAL IS HIGH"); // Send the desired string to convert to speech
      emicSerial.print('
');
   }
while (emicSerial.read() != ':'); // Wait here until the
Emic 2 responds with a ":" indicating it's ready to accept the
next command
}
else{
    emicSerial.print("SIGNAL IS LOW"); // Emic 2 will speak LOW
message if status value is false
    emicSerial.print('
');
    while (emicSerial.read() != ':'); // Wait here until the
Emic 2 responds with a ":" indicating it's ready to accept the
next command

delay(1000); // 1 second delay
}

As the talking logic probe toggles between two phrases, based on the detected
input signals, occasionally, a text phrase can be out of sync. For example, the
SIGNAL IS HIGH phrase may be heard through the speaker with no +5 V DC input
signal detected by the Arduino. This unstable condition occurs when a digital or
microcontroller's assigned pin isn't properly terminated. Electronic circuits such as
a microcontroller's external crystal oscillator can produce radiated switching signals
capable of disrupting its operating embedded code. The unstable speech heard
through the speaker is, again, produced by the crystal oscillator's radiated switching
signal. Removing this electrical noise consists of providing a signal path to ground
using a pull-down resistor. Adding a 1 kilo ohm resistor between the Arduino's D5
pin to ground will provide a stable speech output for the talking logic probe. The
solderless breadboard wiring diagram and circuit schematic diagram show a 1 kilo
ohm resistor added to the Arduino's D5 pin.
A Talking Logic Probe

For reference, the following is the talking logic probe circuit schematic diagram:

If a 1 kilo ohm resistor is not available, a 10 kilo ohm resistor may be used as a replacement pull-down resistor.

Another feature that can be added to the talking logic probe is a momentary pushbutton electrical switch for activating the detection function. With the test lead attached to the digital circuit's specific I/O pin, a press of the momentary pushbutton switch will provide a quick go-no-go check of the device's electrical operation. To include this manual signal detection feature requires the momentary pushbutton switch to be wired between the logic probe's test lead and the Arduino's D5 pin.
This wiring connection is also known as a series circuit. The following is the solderless breadboard wiring diagram for adding the momentary pushbutton electrical switch:
How does the talking logic probe code work

The talking logic probe sketch is modified from the EMIC 2 sample code given earlier in this chapter. The library necessary for the EMIC 2 TTS module to work correctly is:

```cpp
#include <SoftwareSerial.h>
```

Removing this library (header file) from the sketch will cause compilation errors throughout the code, and not allow the Arduino to correctly operate the EMIC 2 TTS module. The pins needed to send digital data between the Arduino and the EMIC 2 TTS module are:

```cpp
#define rxPin   10  // Serial input (connects to Emic 2's SOUT pin)
#define txPin   11  // Serial output (connects to Emic 2's SIN pin)
```
The traditional serial communication pins on the Arduino are D0 and D1. These digital pins may be used in the talking logic probe sketch, but must be removed while uploading the sketch. The reason is because the physical link between the Arduino and the EMIC 2 TTS module while uploading a sketch requires using pins D0 and D1. There is an electrical con
fl
ict in establishing communications between the desktop PC or notebook computer and the EMIC 2 using the true tx:D0 and rx:D1 pins of the Arduino. To illustrate this electrical communication conflict, change the rx and tx pins to D0 and D1 on the Arduino. Upload the modified sketch to the Arduino and notice the following error displayed on the IDE's message window:

To troubleshoot this problem, remove the two wires connected to the Arduino tx and rx pins while uploading. The wires can be reconnected to these pins after the sketch has been uploaded to the Arduino.

The next two lines of code to examine are as follows:

```cpp
#define probein 5  // Digital probe input
int probe=0; // probein variable status
```

The talking probe test lead is connected to pin D5 of the Arduino. The digital signals are received using this pin. The `int probe = 0` status variable will be used to collect the test lead's digital signals during circuit testing. With the test lead pin defined with its signal status variable, the declared Arduino pins operating modes will be set up using the following lines of code:

```cpp
// define pin modes
pinMode(rxPin, INPUT);
pinMode(txPin, OUTPUT);
pinMode(probein, INPUT);
```
A Talking Logic Probe

The `void setup()` function is used to establish the I/O modes for the declared Arduino pins. The next few lines of code help establish the communication protocol (rules) between the Arduino and EMIC 2 TTS module:

```cpp
  // set the data rate for the SoftwareSerial port
  emicSerial.begin(9600);

  emicSerial.print('\n'); // Send a CR in case the system is already up
  while (emicSerial.read() != ':'); // When the Emic 2 has initialized and is ready, it will send a single ':' character, so wait here until we receive it
  delay(10); // Short delay
  emicSerial.flush(); // Flush the receive buffer
```

The preceding lines of code are required when using the EMIC 2 TTS module for different voice/speech synthesis projects. For creating the talking logic probe sketch, a few lines of code were taken from the EMIC 2 sample sketch from the Parallax Inc website. Code reuse, as illustrated here, saves time in developing Arduino sketches and allows the maker to focus on building the electronics instead of debugging software.

The `void main()` code allows the Arduino to read the digital signal detected and speak the voltage level phrases. The first line of code enables the EMIC 2 for speech:

```cpp
  // Speak some text
  emicSerial.print('S');
```

With the text-to-speech module ready to speak, the test lead's status can be read using `probein`, and stored in the variable `int probe`:

```cpp
  probe=digitalRead(probein);// read probein status
```

Once the signal level has been stored in the status variable `probe`, determining which message to speak is based on this conditional statement:

```cpp
  if(probe == HIGH){ // Emic 2 will speak HIGH message if status value is true
    // Code to speak the HIGH message
  }
```
If the statement is true, the EMIC 2 TTS module will speak the phrase `SIGNAL IS HIGH` using the following line of code:

```java
emicSerial.print("SIGNAL IS HIGH"); // Send the desired string to convert to speech
```

Once this text is converting into speech by the EMIC 2, a carriage return is needed to speak the next line of text:

```java
micSerial.print('
');
```

The next action required is to set up handshaking (communications acknowledgement), where the EMIC 2 waits for a new text message to be sent by the Arduino for speech conversion. Here's the line of code required for the Arduino to EMIC 2 TTS module handshaking:

```java
while (emicSerial.read() != ':'); // Wait here until the Emic 2 responds with a ":" indicating it's ready to accept the next command
```

The next message phrase for speech conversion when the signal detected by the test lead is low is `SIGNAL IS LOW`. The else keyword completes the conditional statement instruction of the Arduino code given here:

```java
else{
    emicSerial.print("SIGNAL IS LOW"); // Emic 2 will speak LOW message if status value is false
    emicSerial.print('
');
    while (emicSerial.read() != ':'); // Wait here until the Emic 2 responds with a ":" indicating it's ready to accept the next command
```
A Talking Logic Probe

The `delay (1000)` instruction allows the two phrases to repeat every one second, as shown in the last line of Arduino code. The following flowchart graphically summarizes the talking logic probe sketch. The structure of this flowchart can serve as a template when developing EMIC 2 applications. The final section of the chapter will end with a discussion on DecTalk speech synthesizer engine. Code examples of how to set various feature/functions of the EMIC 2 TTS module will be provided for further hands-on exploration.
The continuation of the flow chart is shown as follows:

A

Read Digital Signal

If Digital Signal is "HI"

Yes

Convert text to speech with EMIC 2

Establish Handshaking: Arduino to EMIC2

Speak "SIGNAL IS HIGH"

1 sec delay

No

Convert text to speech with EMIC 2

Establish Handshaking: Arduino to EMIC2

Speak "SIGNAL IS LOW"

B
DecTalk speech synthesizer engine

The EMIC 2 TTS module uses a speech synthesizer called **DecTalk**. The DecTalk synthesizer is capable of producing a natural sounding voice based on the smallest memory IC footprint. The synthesizer is able to speak in multilanguages, supporting US English, Castilian, and Latin American Spanish. The 12-page document that Parallax Inc developed provides a wealth of information on maximizing the features of the EMIC 2 TTS module. Following are the examples from the Parallax document on how to set the internal feature/functions of the EMIC 2 using the DecTalk speech synthesizer and Arduino code:

- **Vx**: This sets the audio volume in decibels (dB).

  The audio output volume of the EMIC 2 TTS module can be set in the range of 48 dB (softest) to 18 dB (loudest) using the following Arduino code (the default volume setting is 0):

  ```
  // Set volume to x: x= -48 to 18(max volume)
  emicSerial.print("V5\n");
  while (emicSerial.read() != ':');   // wait for ':' character
  ```

  The speech output is sent to a bridged audio power amplifier. A Texas Instruments LM4864 bridged audio power amplifier is used to drive an 8 ohm speaker with an amplification gain (AV) of 10. The LM4864 bridged audio power amplifier circuit is shown here:
With this audio power amplifier IC and the DecTalk volume command, there is no need to add an amplifier circuit to the EMIC 2 TTS module to obtain a suitable output level.

- **Nx**: This selects an EMIC 2 speaking voice. There are nine voices to choose from as listed here:
  1. Perfect Paul (Paulo)
  2. Huge Harry (Francisco)
  3. Beautiful Betty
  4. Uppity Ursula
  5. Doctor Dennis (Enrique)
  6. Kit the Kid
  7. Frail Frank
  8. Rough Rita
  9. Whispering Wendy (Beatriz)

The EMIC 2 TTS module's voice can be changed using the following Arduino code:

```c
// Set voice
emicSerial.print("N1\n");
while (emicSerial.read() != ':');   // wait for ':' character
```

After uploading the modified talking logic probe sketch with the DecTalk set voice command, Huge Harry (Francisco) will be heard through the speaker.

- **Wx**: This sets the EMIC 2 TTS module's speaking rate (words per minute). Depending on the application, the speaking rate may need to be adjusted for clarity and understanding of the message. The range of values that are accepted by the EMIC 2 is 75 (slowest) to 600 (fastest). The default speaking rate value is set to 200. The following is the example Arduino code to set the EMIC 2 TTS module's rate:

```c
// Set speaking rate
emicSerial.print("W100\n");
while (emicSerial.read() != ':');   // wait for ':' character
```
A Talking Logic Probe

- **Lx**: This sets the EMIC 2 TTS module's language.

  The final DecTalk speech command is selecting a language for the EMIC 2 TTS module. The default language is English (0), but two other languages can be selected for the EMIC 2 with the following code:

  Castilian Spanish
  Latin Spanish

  Here is the example code:

  ```
  // Set language
  emicSerial.print("L2\n");
  while (emicSerial.read() != ':'); // wait for ':' character
  ```

  After uploading the modified sketch, the talking logic probe will speak the two phrases in Latin American Spanish.

**Summary**

In this chapter, a discussion on the theory and operation of a talking logic probe was covered. A basic talking logic probe design was shown along with assembly and testing instructions for building the speech synthesis electrical tester. Finally, technical details of how the EMIC 2 text-to-speech module operates were discussed using a series of hands-on experiments. The experiments showed how to set specific EMIC 2 DecTalk speech synthesizer engine feature/functions using Arduino code programming techniques.

In the next chapter, a hands-on investigation on using a web page as a Human Machine Interface (HMI) controller will be illustrated. As in the previous chapters, a series of lab experiments, followed by a final project will show how to build an Arduino-based HMI controller.
Where to buy this book

You can buy Arduino Electronics Blueprints from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.