The creation of the THEREPING

By Vern Graner

Introduction

The Thereping is a musical instrument that requires no specific musical skills to play. It uses a combination of a sonar sensor and a microcontroller to allow the user to play interesting melodies without requiring any musical skills. When joined together with a Master Clock sync unit, the Therepings can all play together and create interesting songs, again without requiring any musical expertise on the part of the players.

In the beginning

The quest for a new and interesting musical instrument began when the Robot Group of Austin (http://www.robotgroup.net) was selected to participate in the First Night Austin celebration (http://www.firstnightaustin.org) with a proposal they entitled "The Robot Theremin Band". The idea was to use Theremins (see sidebar: "Whats a Theremin?) and Robots together to create an experimental, musical spectacle.

This project was discussed through the group's mailing list and I happened to mention that a group of technically minded folks with no musical training trying to play Theremins would sound roughly like someone attempting to murder a large number of cats with a mallet. Since this might be more apt to *repel* people than attract them to our presentation, I suggested that we might consider crafting some sort of electronic musical instrument instead. The topic received little more attention as the event seemed rather far away ("seemed" being the key word).

Inspiration

Later, at a meeting held at my house, Don Colbath (one of the group members who actually owns a *real* Theremin) noticed I had a Parallax Basic Stamp connected to a Parallax PING))) sonar sensor (Figure 3). The test system was running a program that would show the distance of an object from the PING))) sensor on an LCD display. Don remarked that if the distance reading could be sent to an oscillator, it might be

What's a Theremin?

The theremin or thereminvox (originally pronounced [teremi:n] is one of the earliest fully electronic musical instruments. Invented in 1919 by Russian Léon Theremin, the theremin is unique in that it requires no physical contact in order to produce music and was, in fact, the first musical instrument designed to be played without being touched. The instrument consists of a box with two projecting antennas around which the user moves his or her hands to play. To control the theremin, the musician stands in front of the instrument and moves his or her hands in the proximity of two metal antennas, the distance from the antennas determining frequency (pitch) and amplitude (volume). Small movements of the hands can create a tremolo or vibrato effect. Typically the right hand controls the pitch and the left hand is used for the volume, although some play left-handed. Based on the principle of heterodyning oscillators, the theremin generates an audio signal by combining two different but very high frequency radio signals. The capacitance of the human body close to the antennas causes pitch changes in the audio signal, in much the same way that a person moving about a room can affect television or radio reception. By changing the position of the hands relative to the vertical antenna, a performer can control the frequency of the output signal. Similarly, the amplitude of the signal can be affected by altering the hand's proximity to the looped antenna. The information above is excerpted from "Wikipedia", located at the following URL: http://www.wikipedia.org

possible to make a sound similar to a Theremin! Intrigued, I wrote some very simple PBASIC code that would measure the distance reported by the sonar sensor in Milliseconds, and store it in a variable. Then, rather than try to coble together an oscillator, I just stuffed that value into the PBASIC command used to create sounds, the aptly named "FREQOUT" to have the basic stamp produce sound directly as shown here:

Figure 2: Gray Mack and the Prototype Thereping

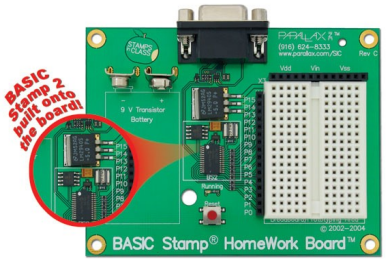
```
PING     PIN 4    ' Parallax PING)))
SPEAKER  PIN 9    ' Speaker
SAMPLE   VAR Word ' Store result

AGAIN:
  PULSOUT PING, 5
  PULSIN PING, 1, sample
  FREQOUT SPEAKER, 100, sample
GOTO AGAIN
```

When we ran the above code, the unit would play sounds at a pitch in proportion to the distance of your hand to the sensor! We had a basic proof of concept for a new musical instrument. The "Thereping" was born!

Parts is Parts

Now that we had a proof of concept, it was time to order in the components so we could begin to fabricate a bunch of instruments. I spoke with Denise Scioli, the robot group member who had been in close contact with the First Night folks, to see if we had the budget to purchase enough parts to get started building these devices. She verified we had been funded and she was able to contact Parallax and order in all the parts we needed. Since the prototype instrument was based around the Basic Stamp II microcontroller from Parallax, we decided on the Parallax "Homework Board" (Figure 1) to act as the base for all the instruments. Denise placed an order and soon we were in possession of ten "Homework Boards" and ten PING))) sonar sensors. In preparation for the project, Denise had created and decorated some wooden "cutouts" in fanciful shapes of rocket ships and robots that were to be used as a base to hold the homework boards and provide a "familiar' playing paradigm (i.e. "guitar like"). I attached one of the homework boards and a PING))) sensor to one of the boards to get a feel for how the instrument would look and feel (Figure 2).

Figure 1: The Parallax Homework Board



Since we were planning on making a "band" of instruments, we decided the best approach would be to buy identical components. Subsequently we chose to purchase a "10 pack" of Parallax Homework boards. These are self-contained Basic Stamp 2 microcontrollers with a built in Breadboard and DB9 serial interface for programming. The Homework Boards are only sold in a ten or twenty piece pack, but this would give us enough units to have six instruments, one "master sync" clock and a few "spares" to use for experimentation and as backups in the event of a component failure. At quantity ten, the per-board price drops to only $40.00.

Refining the idea

Though we had one prototype circuit and mount for our fledgling Thereping, and this prototype would make sound, it was not necessarily a *pleasant* sound. Also, like an actual Theremin, it required some amount of skill in music and physical control to create consistent musical notes and/or melodies. More importantly, if we built a number of these units, when played together they would again suffer from the same weakness a group of "real" Theremins would have (i.e. the "malleted felines" issue). I decided to break the problem into pieces and solve the problem one piece at a time.

What can go wrong?

For our purposes, I decided that there are two *fundamental* things that you can do "wrong" when playing an instrument:

1) You could play an incorrect note
2) You could play a note at the wrong time

I decided to attack the "incorrect note" issue first. I knew there were scales of notes where any of the notes in the scale, when played together, would harmonize. I decided to use a "blues" scale in the key of C for my experiment. On the prototype instrument, I created code that would instruct the PING))) sensor to fetch a distance reading, and then would check the returned reading against a range of values to determine what note should be "played" by the basic stamp. The first code looked like this:

```
AGAIN:
  PULSOUT PING, 5
  PULSIN PING, 1, sample
' C blues scale      C - Eb - F - Gb - G - Bb - C
IF SAMPLE1 > 1000 AND SAMPLE1<1047 THEN FREQOUT SPEAKER,100,1047 ' C
IF SAMPLE1 > 1047 AND SAMPLE1<1245 THEN FREQOUT SPEAKER,100,1245 ' Eb
IF SAMPLE1 > 1245 AND SAMPLE1<1396 THEN FREQOUT SPEAKER,100,1396 ' F
IF SAMPLE1 > 1396 AND SAMPLE1<1480 THEN FREQOUT SPEAKER,100,1480 ' Gb
IF SAMPLE1 > 1480 AND SAMPLE1<1568 THEN FREQOUT SPEAKER,100,1568 ' G
IF SAMPLE1 > 1586 AND SAMPLE1<1864 THEN FREQOUT SPEAKER,100,1864 ' A#/Bb
IF SAMPLE1 > 1864 AND SAMPLE1<1976 THEN FREQOUT SPEAKER,100,1976 ' B
IF SAMPLE1 > 1976 AND SAMPLE1<2093 THEN FREQOUT SPEAKER,100,2093 ' C7
GOTO AGAIN
```

Though not very *efficient*, the above code did brute-force determine the sonar distance and then make it so the instrument would only play the notes that would be valid in this blues scale. When this code is run, you can move your hand above the sensor and hear all the notes in the blues scale. However, the timing between the notes is fixed at 100ms (roughly 16th notes at 120bps) and your hand position is absolute (i.e you cannot change the distance your hand must travel to create specific notes).

Figure 3: PING))) Sensor

The Parallax PING))) sensor uses ultrasonic sound waves to determine the distance to an object and returns a value in milliseconds that is quite accurate. In addition it is very easy to use and connect since it only requires three connections, 5v+, GND and a single TTL level pin for both sending and receiving data. The Parallax website offers the these units in a "five pack" so it was possible to leverage quantity purchase prices and buy two 5-packs of PING))) sensors to match the one ten pack of Homework boards. This brought the price per sensor down to about $20.00.

At the next Robot Group meeting, I showed the code to Eric Lundquist, another long-time Robot Group member (and a programmer by trade), and together we refined the code in a number of ways to make the instrument more efficient and flexible. First, it was decided to break the space above the PING))) sensor into discrete "zones' that could be "calibrated" to not only make the sensing area adjustable, but to also allow the notes to be scaled to different octaves or pitches. This was accomplished by assigning the highest and lowest PING))) sensor readings acquired while comfortably moving our hand over the sensor (from a bit less than 1" to about 8") and setting those values into constants as shown here:

```
LOWPos  CON 100   ' MS Value for closest note
HIGHPos CON 800   ' MS Value for furthest note
```

Then we set a constant to hold the Number of Notes we wanted into which we wanted the range divided as shown here:

```
Nnotes  CON 7    ' Number of notes in the scale
```

This created seven 1 inch zones above the sensor. Now you could tell in

which “zone” the user's hand was detected by using this simple formula:

```
Zone = (HighPos-LowPos) / NNotes
```

In order to make the code easier to read and more intuitive, we used the PBASIC “CON” command to create an “alias” of the Hz values for each note in a 12-note chromatic scale. This alias would reflect the note's “name”, it's modifier (sharp or flat), and the octave. The naming convention would be *<note letter><sharp/natural><octave>*. For example, the notes from C6 through C7 would look like this:

```
Cn6  CON 1047
Cs6  CON 1109
Dn6  CON 1175
Ds6  CON 1245
En6  CON 1319
Fn6  CON 1397
Fs6  CON 1480
Gn6  CON 1568
Gs6  CON 1661
An6  CON 1760
As6  CON 1865
Bn6  CON 1976
Cn7  CON 2093
```

Figure x: Vern Graner and Mike Scioli work on the prototype Thereping

Though we did consider using the Basic Stamp itself to calculate the Hz values for each note in real time, it seemed cumbersome and would require additional processing power that we didn't know if we would need later for other purposes, so we instead just pre-calculated the values. Also, the original “brute force” method of determining the appropriate note to play was discarded in favor of a more elegant approach using the PBASIC “LOOKUP” command to pick a value from a range. The new code worked like this:

1. Check the distance to the hand above the PING sensor

```
PULSOUT PING, 5              ' Send a ping out
PULSIN  PING, 1, sample      ' store response in "sample"
```

2. Determine in which “zone” the hand was located and store it in “note” variable

```
Note = (SAMPLE - LowPos) / Zone
```

3. Use the note number to LOOKUP the correct note frequency and store that value in the FREQ variable

```
' C blues scale      C - Eb - F - Gb - G - Bb - C
LOOKUP NOTE, [Cn6, Eb6, Fn6, Gb6, Gn6, Bb6, Cn7], FREQ
```

4. Play the note with the FREQOUT command.

```
FREQOUT Speaker,100,FREQ
```

After step 4, just go around in a loop and do it all again. Using this new code, we had an instrument that would efficiently play only the correct notes in a specific scale. However, at this point we had no control of the note's duration or of its timing in relation to other instruments.

Plays well with others


Figure : Thereping Band ready to play in the parade!

Since the point of the project was to have everyone play *together*, we needed to attack the second problem: SYNC. Originally, I considered using an LED IR beacon as a sync source for the Thereping instruments. This would consist of a tall tower (a "tempo tower") that would allow IR LEDs to be pointed downward to the stage where the instruments would be during a performance. Since our stage was to be a trailer in a parade, and a small outdoor pavilion, this solution at first appeared to be feasible. However, we discovered from the schedule that we were slated to play on the pavilion in the daytime where natural UV/IR light would drown the sync from any IR LED sync source. It was also pointed out that the musicians would have to have a constant and uninterrupted line of site to the "tempo tower" if continuous sound was to be maintained. If the players were to move about their bodies may block the view to the tower. With these problems, it became clear we needed to rethink our SYNC.

Getting Wired

Though a "wireless" form of sync signal would be beneficial for the artists (no wires to tangle up) it would also require that each instrument contain an on-board audio amplifier, speaker and power source. These would have to be fairly hefty amps and speakers to be heard in the middle of a parade and would require some hefty battery sources, not to mention the additional cost and construction time. Since we had planned to have a central "P.A." system it made sense to simply fall back to a "hard wired" approach. Since it looked like hard-wiring would be a good choice since it solved a number of problems, I created an experimental "master clock" using one of the Parallax BSII Homework Boards" and wrote some simple code to toggle one of the lines HIGH then LOW. The code looked like this:

```
AGAIN:

HIGH 15
PAUSE 250
LOW 15
PAUSE 250

GOTO AGAIN
```

I added an LED to PIN 15 on the breadboard of the Master Clock board so I could verify the code was indeed toggling the output, then I connected a jumper from PIN 15 on the "clock" stamp to pin 15 on the Thereping prototype. Once I had this complete, I modified the Thereping code to include a pin definition for "SYNC" and then added this new line of code just above the existing sound-producing command shown here:

```
SyncWait:
IF SYNC = 0 THEN SyncWait

FREQOUT Speaker,100,FREQ
```

I powered up the Thereping unit, placed my hand over the PING sensor, and the notes began to fire off in precise accordance with the blinking of the LED on the master clock! We had SYNC!

Play it again... And again... and..

Now that we had the prototype master clock and the prototype Thereping circuit on the bench for a bit, I began to notice it was a bit fatiguing to hear the same exact note *duration* without *variation*. Though it was interesting to hear for a short period of time, it occurred to me that a group of these instruments would sound very similar and it might not be possible to determine which instrument was playing and even if you could, the sound produced could be mostly redundant. I thought it would add interest to the performance if the performer could choose the duration of the notes.

For example, play 1/8th notes for a while, then play ¼ notes then switch to 1/16th notes when ever they chose. In order to check for the selection of the performer, some buttons would have to be added. I added three buttons to the Thereping Prototype with the intention of using code to determine if the buttons were pressed and then alter the "duration" of the sound played.

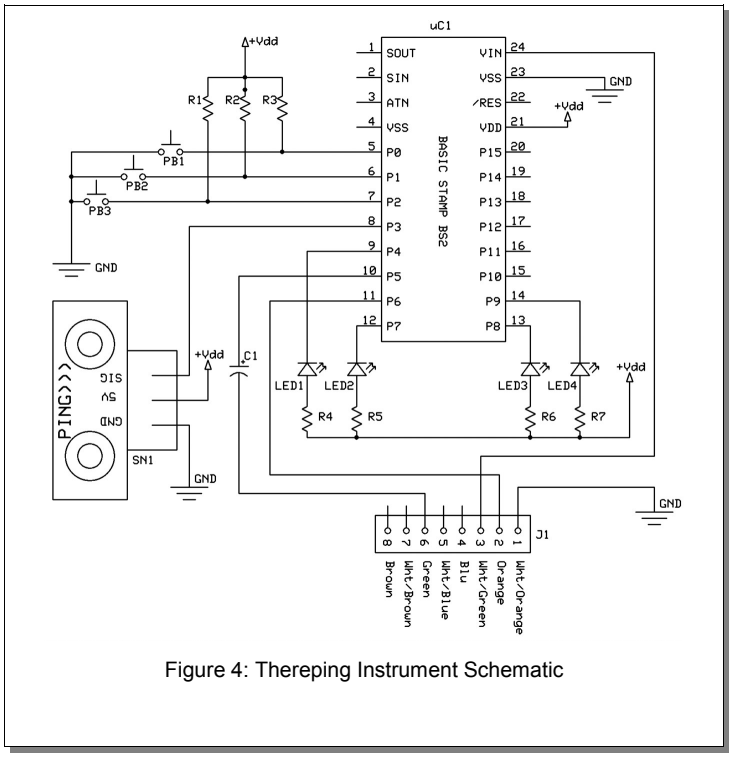


Figure 4: Thereping Instrument Schematic

When I did this, I immediately noticed that what resulted was only the ability to change the note from "filling" the entire ¼ note interval, or to play a 1/8th note followed by an 1/8th rest. In order to have TWO 1/8th notes play, I would have to play the note two times per trigger pulse. This was quickly getting complicated, and we were also quickly running out of time. I decided that if I knew the interval of the clock pulses coming from the Sync Controller, I would be able to play multiple notes before looking to the sync input for a new clock pulse. I altered the code to play the note 2 times for an 1/8th note and four times for a 16th note. This allowed me to "spice up" the playback and allowed for some much more interesting "solos" on the instrument.

Droning on and on...

Now that we had the ability to play a fairly intricate "solo" on the instrument, we were again faced with the problem that many instruments all playing similar sounds would tend to blend into just so much noise. Being exposed to modern American musical culture, most people expect certain things from "music". They expect a tempo, a beat an accompaniment and some sort of melody. At this point all I had was a group of instruments that could play melody or "lead" parts. In order to create a more appealing sound, it was decided that when the player of the instrument was "done" playing their "solo" or "lead" part, they should be able to drop into the background and just enjoy what others were playing. Since we were in the same key throughout the song, I knew there were certain notes that could be played that could act as a foundation to the music. In thinking on this problem, I came across an interesting web site on the design behind the Bagpipes. The Bagpipe is an interesting instrument in that it has a three "horns" that play a consistent chord of notes throughout every song, referred to as "drones". Two "tenor" drones and one "bass" drone. Throughout any bagpipe performance these three

horns play a continuous three note chord. It occurred to me that I could create a sort of “drone” for the instrument using one of the buttons to tell the microcontroller that it should simply play a “drone” and ignore the sonar sensor altogether. This would let a player rest their hand, dance, look around or listen to others playing. This would also allow the instruments to play something closer to what most people expect to hear in modern music. I added code that would detect the “drone” button, and would then branch to a routine in the code that would simply repeat a series of notes in a bass line. The “drone” addition to the code is shown here:

```
DO WHILE BTN1=0  ' Check for "drone" button and play drone sound if pressed
  SyncWait1:
  BUTTON SYNC, 1, 255, 0, BtnWrk, 0, SyncWait1 ' Wait for a sync pulse

  Duration = Clock
  FREQOUT Speaker,Duration,Cn4/2, Cn4 'Play the note!
  PAUSE offset
  Duration = Clock
  FREQOUT Speaker,Duration,Cn4/2, Cn4 'Play the note!
  PAUSE offset
  Duration = Clock
  FREQOUT Speaker,Duration,Cn4/2, Cn4 'Play the note!
  PAUSE offset
  Duration = Clock
  FREQOUT Speaker,Duration,Cn4/2, Cn4 'Play the note!
  PAUSE offset
  Duration = Clock
  FREQOUT Speaker,Duration,Cn4/2, Cn4 'Play the note!
  PAUSE offset
  Duration = Clock
  FREQOUT Speaker,Duration,Cn4/2, Cn4 'Play the note!
  PAUSE offset
  Duration = Clock
  FREQOUT Speaker,Duration,Cn4/2, Cn4 'Play the note!
  PAUSE offset
  Duration = Clock
  FREQOUT Speaker,Duration,As4/4, As4/2 'Play the note!
  LOOP
```

One thing to notice is that the note values are divided by 2 in order to drop the voice of the instrument by an octave (i.e. Cn4/2). This created a “driving” bass line to accompany the other players who would be “soloing”. I created a couple of variations on this “drone” part in order to “change up” the sound. I wanted as much variety in the final song as we could get. You may also notice this is a pretty “sloppy” way to accomplish this task. It would be much tidier to make it a loop, however, were were running out of time before our debut and most of the coding was done with an eye towards illustrating the function rather than optimizing form. Remember this if you decide to build your own Thereping, there are LOTS of opportunities for improvement!

CAT-5 to the rescue!

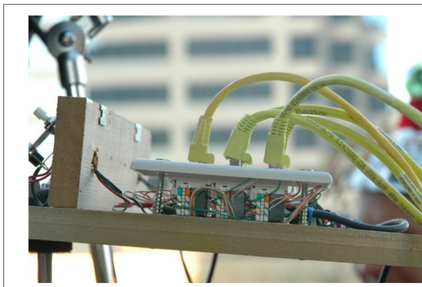


Figure: Master Sync Unit in the parade

Since we had a proof of concept for the syncing of multiple instruments using wire, I now had to look at finding wiring methods to use that would easily and robustly link all the instruments to a central clock. It just so happened that I had quite a few Ethernet cables and RJ45 “keystone” jacks laying about so I decided to try using regular CAT-5 cable to connect the Master Sync unit to the Thereping instruments. The four pairs in the CAT-5 cable would allow me to provide a path for all the needed signals as shown here:

Pin	Color	Use
1	White/Orange	GND
2	Orange	SYNC
3	White/Green	V+
4	Blue	N.C.
5	White/Blue	N.C.
6	Green	AUDIO
7	White/Brown	N.C.
8	Brown	N.C.

Also, CAT-5 cables are easy to come by in many lengths and colors, and there would be enough wires to allow for future expansion if the need arose. The final schematic for the Thereping shows the CAT-5 connector as the sole I/O point for the instrument providing everything that is needed to operate the instrument (Figure 5).

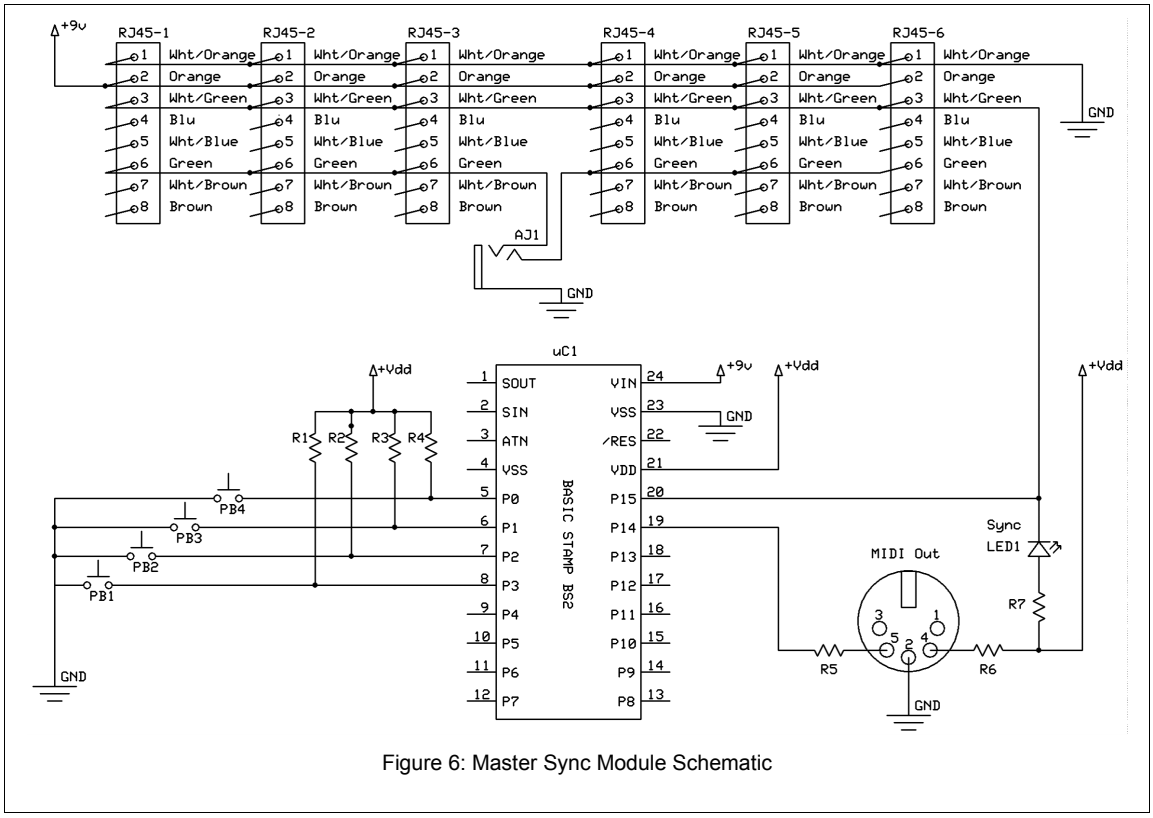


Figure 6: Master Sync Module Schematic

Gimme a BEAT!

Ok, so we had a lead, and we had a bass line, and we had a way to keep all the instruments in sync, the only thing that was missing in my mind was a BEAT. I did some research on the MIDI protocol and discovered that, though the Basic Stamp did not have the speed to reliably receive MIDI data (no serial buffering), it did have the ability to *send* MIDI, data easily. I decided it

The master clock in its finished format mounted on a Yamaha DTXpressII drum kit: