Arduino Bat Simulator

I've been messing around with ultrasound listening and bat detector circuits, but often there are no obliging flying critters to talk to me! You can generate ultrasound in many ways (rubbing fingers, dangling keys, rushing water) but I wanted a reliable bat-like sound source, so I wrote some simple Arduino code.

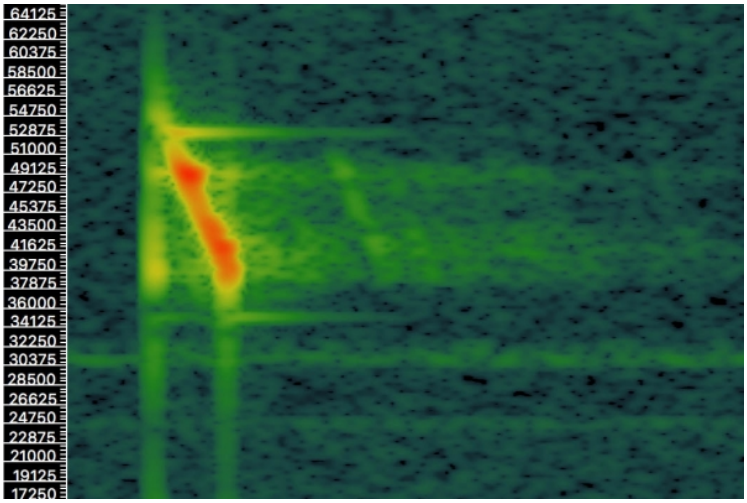
First we need to manipulate the Arduino timers to reliably generate higher frequencies. This forum thread describes a clever technique. There is no DAC on an Arduino so we are limited to square-waves, but I don't mind some harmonics!

Hardware

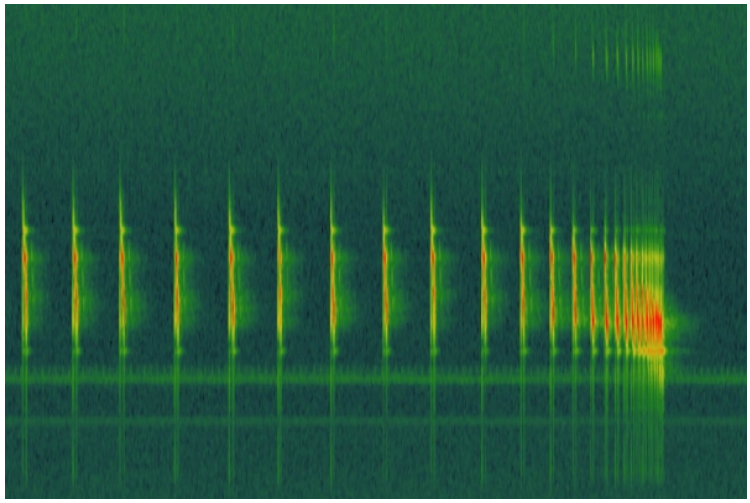
- I used a 3.3v Arduino Pro Mini clone since they are cheap, small, and reasonably power efficient.
- Power is provided by a 30xx lithium battery, via a slide switch.
- The ultrasound emitter is a 16mm 40kHz transducer, popular in sonar proximity sensors.
- Wire the transducer (+) to Arduino pin 9 via a 100 ohm resistor. Transducer (-) should be grounded.

Software

The code generates a chirp sequence with slightly randomized pitches and intervals, then pauses before repeating. Occasionally a sequence will end with a sub-sequence of rapid clicks, simulating the FM feeding buzz as the bat closes in on an insect. (The onboard LED blinks with each chirp or click, since you can't hear any of this with your ears!) The result is not the cleanest waveform, but it translates quite well when testing bat detector hardware.



Single chirp (screenshot scale is 1/10 actual frequency)



A chirp sequence followed by feeding buzz

Copy/paste the code below or **Download bat_simulator.ino (zipped)**

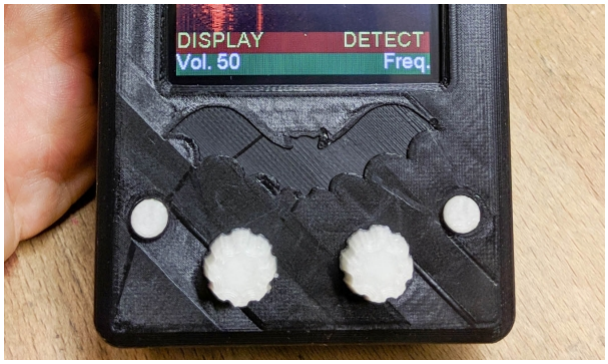
```
/* ----------------------------------------------------------------------
 *  FAKE BAT SIMULATOR for testing bat detector hardware. - Zach Poff - 2019
 *  Attempts to emulate "hockey-stick" sliding chirps and quick FM hunting clicks
 *  square-wave generator adapted from https://forum.arduino.cc/index.php?PHPSESSID=r994n5acmnip4de52nf2qoc6c2;topic=116094.msg8
 *  Attach 40kH piezo transducer between GND & pin 9 in series with 100ohm resistor
*/

#include <avr/io.h>
#include <avr/interrupt.h>
int LEDpin = LED_BUILTIN;

void setup(){
  pinMode(9,1);
  pinMode(10,1);
  Serial.begin(115200);
  Serial.println("Bat Simulator - Zach Poff 2019");
  pinMode(LEDpin, OUTPUT);
}

void loop() {

  //a sequence of normal chirps
  int numchirps = random(10, 20);
  Serial.println(String(numchirps) + " chirps");
  for (int count = 0; count <= numchirps; count++) {
    batChirp(random(55000, 57000), random(40000, 42000));
    delay(random(70, 85));
  }
  //every now and then, do a sequence of "FM" hunting clicks after a sequence
  if (random(0, 3) == 0) {

    float clickdelay = random(70, 85);
    long clickHigh = 56000;
    long clickLow = 42000;
    int numclicks = random(10, 18);
    Serial.println(String(numclicks) + " clicks");
    for (int count = 0; count <= numclicks; count++) {
     batChirp(clickHigh, clickLow);
     delay(clickdelay);
     // shrink the delay time and frequencies each time (constrained to sane values)
```

```
    // shrink the delay time and frequencies each time (constrained to safe values)
     clickdelay = constrain(clickdelay * 0.75, 0.1, 85);
     clickHigh = constrain(clickHigh * 0.98, 40000, 65000);
     clickLow = constrain(clickLow * 0.98, 30000, 65000);
     //Serial.println("ClickDelay: " + String(clickdelay) + " High: " + String(clickHigh)+ " Low: " + String(clickLow));
     }
  }

 //pause between sequences
  delay(random(2000, 6000));
}

// chirp function args: start freq(hz), end freq(hz)
void batChirp(long startF, long endF) {
  int stepSize = 400; // Hz per step of first loop (gets smaller during later loops)
  digitalWrite(LEDpin, HIGH);
  DFGon();
  for (long freq = startF; freq >= endF;) {
    DFGfreq(freq); // Play the frequency sweep
    stepSize = (stepSize * .994); // stepsize shrinks with each loop to approximate a "hocky stick" log pitch envelope
    freq = freq - (stepSize + 1); // BUT it will never reach endF (since it decreases by a fraction of itself) so we add 1
    //Serial.println(freq); //debugging slows things down
  }
  DFGoff();
  digitalWrite(LEDpin, LOW);
}

// "DFG" is a dynamic function generator (precise squarewave using timers)

void DFGon(){
  cli();//disable interupts
  TCCR1A = 0;//registers for timer 1
  TCCR1B = 0;
  TCNT1=0;
  TCCR1A |= _BV(COM1A0) + _BV(COM1B0);
  TCCR1B |=_BV(WGM12);
  TCCR1C = _BV(FOC1A);
}

void DFGoff(){
  TCCR1A &= ~(1 << COM1A0);
  sei();  //enable interupts
}

 void DFGfreq(unsigned long tempfreq){
  tempfreq = tempfreq*2;
   if(tempfreq > 122 && tempfreq < 1000001){
  OCR1A = (8000000/tempfreq)-1;//#TIMER COUNTS
  TCCR1B |= _BV(CS10);
  }
  else if(tempfreq <= 122 && tempfreq > 15){
    OCR1A = (1000000/tempfreq)-1;
    TCCR1B |= _BV(CS11);
  }
  else if(tempfreq <= 15 && tempfreq > 4){
    OCR1A = (125000/tempfreq)-1;
    TCCR1B |= _BV(CS10) + _BV(CS11);
  }
}
```

Further Research

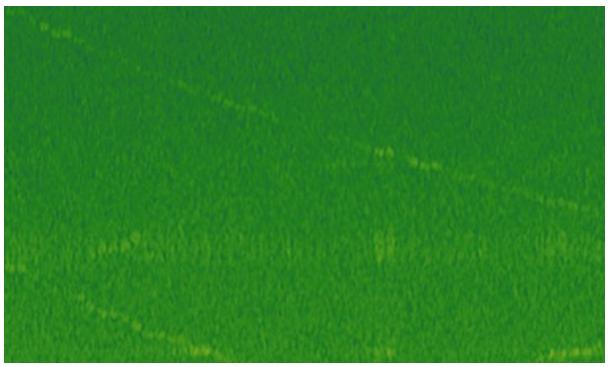
- Larry Jones' **Bat Amigo & Bat Buddy** (and a mention of Tony Messina's **Bat Chirp**)
- The **PiBat Bat Simulator** is another Arduino-based approach.
- The **microbat board** from the German "bats in Kiel" blog

Related Posts

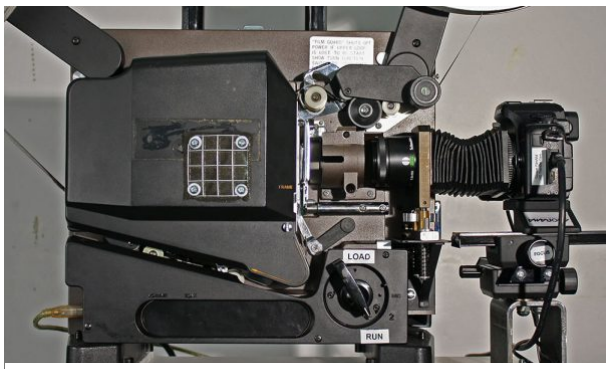


DIY "Teensy" Bat Detector

Exploring Ultrasound

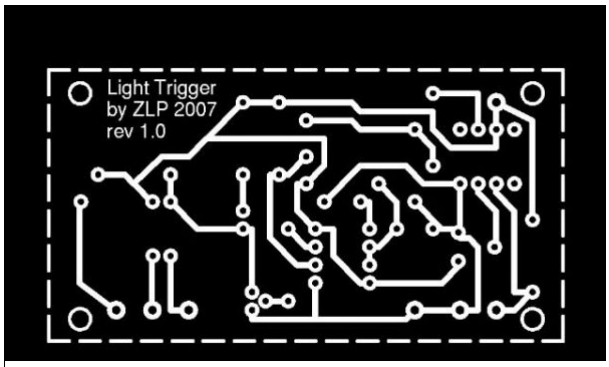


Cheap Microphones for Ultrasound



16mm Film to HD Transfer System

Audible 16mm Footage Counter



Light Trigger Circuit