Music Synthesizer

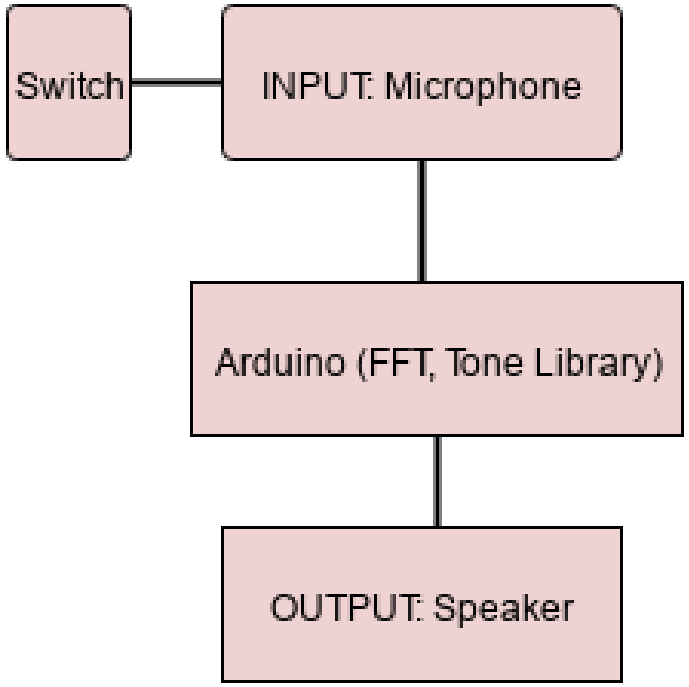
Neil Baker

Electronics Lab

Overview

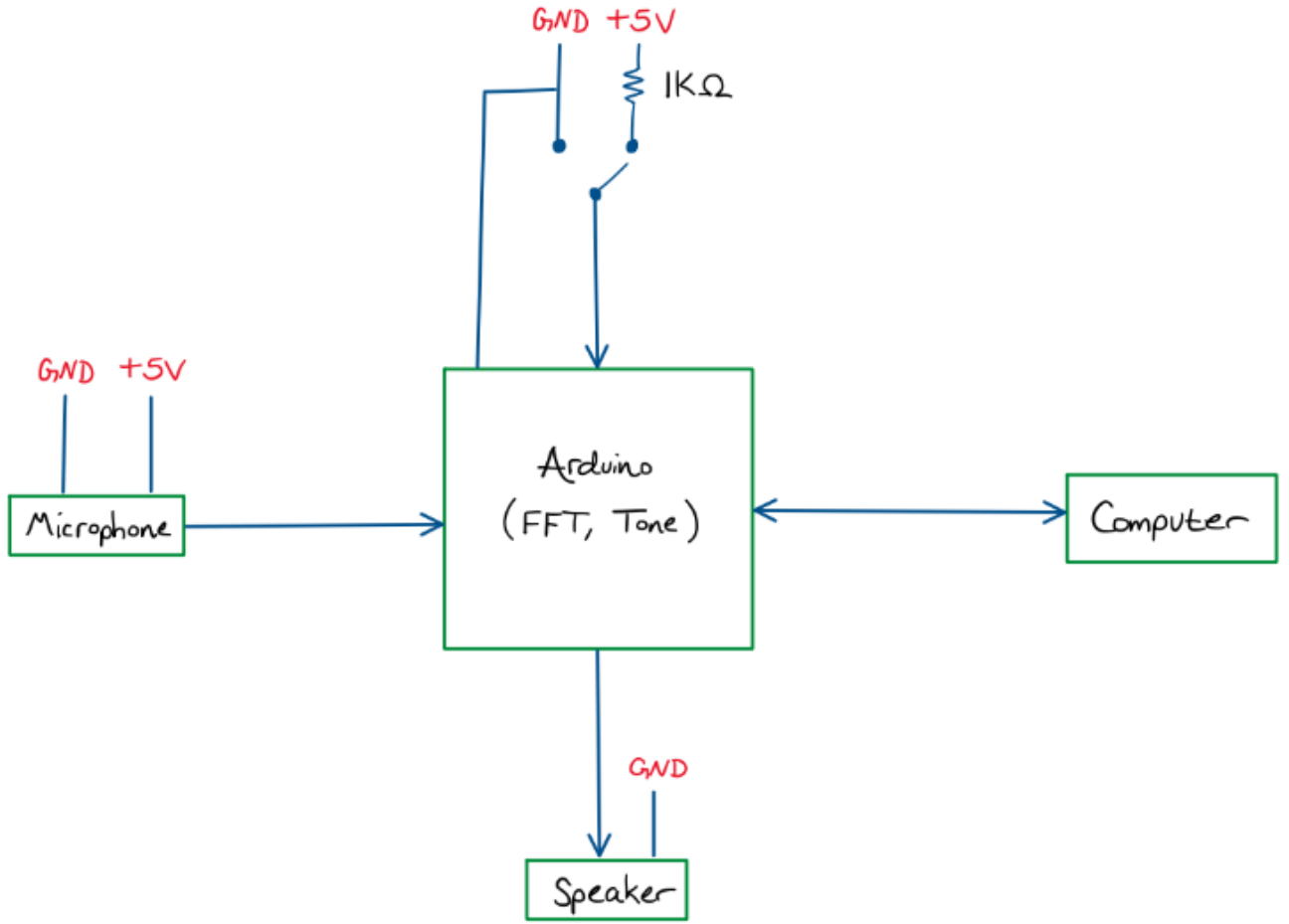
The music synthesizer takes a steady tone through a microphone, auto tunes it, and plays the auto tuned note through a speaker, along with embellishments. You can have it play an arpeggio using the sung note as a root, or any number of other things. The sampling rate is about 1000Hz and the speaker reproduces the tone with some accuracy (usually within two semitones).

Design



Analogue to Digital

Digital to Analogue



Block Diagram Breakdown

- Switch (decide when to take samples)
  - On/off
  - Print statements tell user when to start/stop singing
- Microphone (take samples)
  - Take samples for ~1 second
  - Sampling rate ~1000 Hz
- Arduino microprocessor (FFT and Tone Library)
  - Fast Fourier Transform (FFT) function converts amplitude data into frequency
  - My program takes that frequency, auto tunes it and embellishes it with other notes
  - Tone library converts
- 8Ω Speaker
  - Play note from analogue signal sent by Arduino

Some Code

```
void loop() {

  if (digitalRead(switchPin) == HIGH) {
    Serial.println(1);
    delay(1000); // begin singing
    // collect data over 1 second
    for (int i=0; i<1024; i++) {
      micVal = analogRead(micPin);
      micVals[i] = micVal;
      //Serial.println(micVals[i]); // use to see all micvals
      delay(1);
    }
    Serial.println(0);
    float f=Approx_FFT(micVals, 1024, 1000);
    // Serial.println(f); // test to see if FFT function returns
    int a=autoTune(f);
    playArp(a);
    //tone(8, f, 1000); // test
    delay(5000); // in this time, flip the switch to low
  }
```

Some More Code

```
/* a function that takes the frequency outputted by the FFT function and returns
the frequency of the note that is closest to the input frequency */
int autoTune(int f) {
  int note = 62; // baseline set to B1

  for (int i=1; i<73; i++) {

    if (f < notelist[i]) {
      int below = notelist[i-1];
      int above = notelist[i];

      int subtractBelow = f - below;
      int subtractAbove = above - f;

      if (subtractBelow < subtractAbove) { // see which note the sung note is closest to
        note = below;
        // tone(8, note, 100);
        return note;
      }
      else {
        note = above;
        // tone(8, note, 100);
        return note;
```

```
// start at B1 and go to B7
int notelist[73] = {62,65,69,73,78,82,87,93,98,104,110,117,123,131,139,147,156,165,175,185,196,208,220,233,247,262,277,294,311,330,349,370,392,415,
```

Results

- The switch is effective in limiting the sampling time and giving the user a well-defined time over which to record
- The synthesizer always recognizes the input signal
- The FFT function is able to reproduce the tone with some accuracy, usually within about a tone.

Limitations

Arduino memory limitations prohibit me from taking more than a thousand samples through the microphone. Since the microphone sampling rate should be much higher than the tone frequency in order that the FFT function accurately measure the tone frequency, the duration over which we take samples must be shortened. I am now taking 1024 samples over about one second, and it is still not particularly accurate.

It is unclear whether the hardware capabilities include playing more than one note at a time. If I were to continue the project, I would have the program play intervals and chords. More work required to see if this is feasible given this setup.

Thank You!