ANALGITAL SYNTHESIZER

WORKSHOP DONE BY PECHBLENDA 11-02-2014

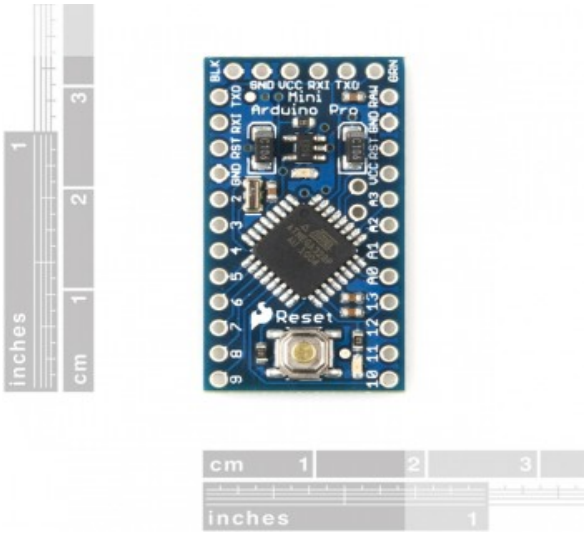
→ pechblenda.hotglue.me (Bio-electro-chemical transhackfeminist laboratory)

→ network23.org/pechblendalab

→ calafou.org (place where we live and have the laboratory; come and visit!)

Our lover Arduino pro miNi

Arduino Pro Mini is a compact but powerful Arduino board.



This tiny little board was born early on Feb 2012. You can get more info here:

- ATMEGA 328 and pre-loaded with Arduino Duemilanove Bootloader
- A reset button
- 5V voltage regulator, reverse polarity protection
- 16MHz Crystal
- On board Power LED indicator
- A programmable LED (Status LED), D13 as on Arduino board.
- Pin label is compatible with Arduino

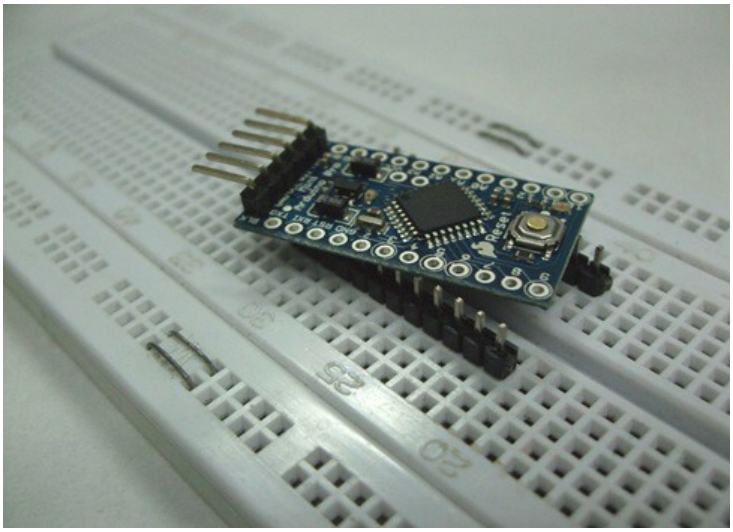
The pad is extended out, you can choose to solder header socket, straight header pin, Right angle header pin, turn pin, etc. Totally up to you

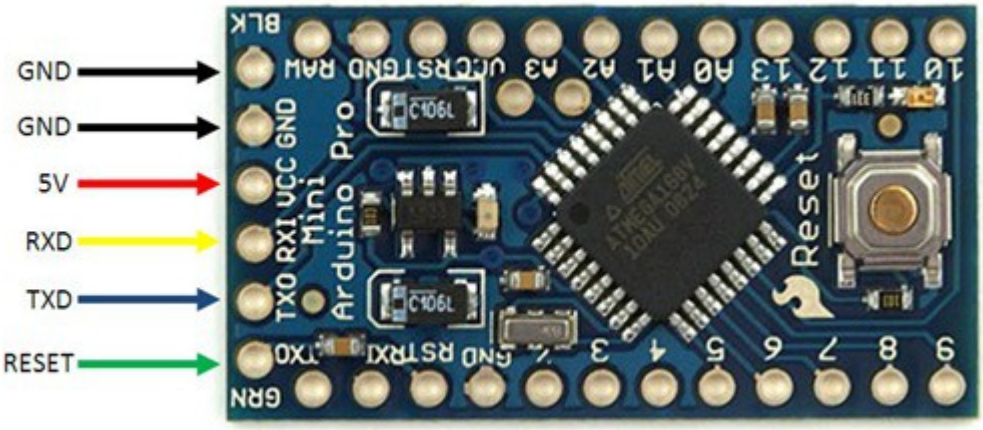
Be creative.

always is nice to know about the schematics:

this is the 02.schematics into our folder AnalGytal synth.

Arduino PRO Mini Bootloader pin





Installation

http://www.arduino.cc/en/Main/Software

First we start installing the Arduino IDE program on your computer. It is available for Mac, Windows and Linux. We download it from .

Depending on the model of Arduino board and computer you are using, sometimes you'll need to download FTDI drivers.

Arduino IDE is a really nice and easy platform, very intuitive and there is a lot of information on the net.

Most arduino boards come with USB-to-serial converter to connect the board by usb to your laptop. However, Arduino pro mini (the board we are using) does not have USB-to-serial converter soldered on the board, so you'll need to use the FTDI USB-to-serial driver chip as a connector.
https://learn.sparkfun.com/tutorials/how-to-install-ftdi-drivers/all

Once we have Arduino IDE installed and Arduino Pro Mini connected to the computer through the FTDI USB-to-serial driver chip, we can start playing with code and examples (that this platform offers).

First we have to check if Arduino IDE is detecting the USB port by going to Tools-->Port and make sure we see a line with the text **/dev/cu.usbserial-xxxxx** where the xxx's can be anything. Same for **/dev/tty.usbserial-xxxxx**. This indicates that the driver is installed properly and that the Arduino was found.

Next, we have to choose the type of board of Arduino we are using. So we go to Tools-->Board and select the board we are using. In our case, we'll choose Arduino Pro or Pro Mini option. Sometimes this option does not exist, if this happens then we will choose Arduino Nano option, which is the equivalent.

Once everything is selected, we can compile and upload our code correctly.

2. Inside Arduino IDE you can find plenty of examples and code to try. We will start with a simple code by blinking a led.
   http://www.ladyada.net/learn/arduino/lesson2.html

Programming on arduino ide is easier than it looks. Many tutorials and coding can be found on the internet. On arduino.cc webpage you can find helpful information and on Reference category http://arduino.cc/en/Reference/HomePage you can find the Language Reference for programming with arduino.

The Arduino environment can be extended through the use of libraries, just like most programming platforms. Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. A number of libraries come installed with the IDE, but you can also download or create your own.

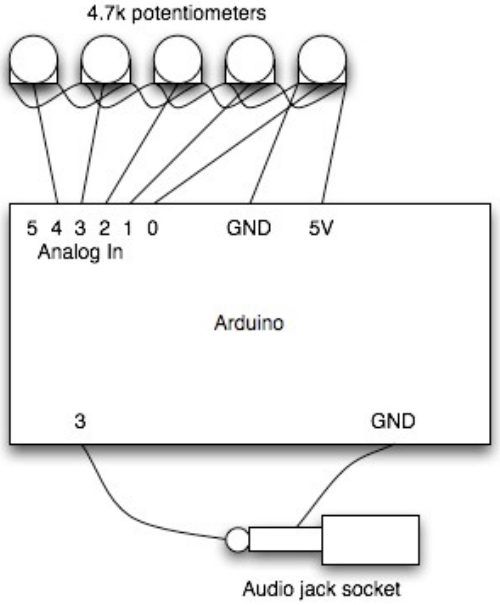
3. Analgital Synthesizer based on Auduino Synth by Tinkerit.
   http://code.google.com/p/tinkerit/wiki/Auduino

in 2008 tinkerit developed this synthesizer and opened the code and hardware on the net. The code is done so that is able to be use with several boards and with several options.

Auduino works with 5 potentiometers, a jack output and some lines of code that can be implemented.

It's a granular synth that mixes analogic functions, such as potentiometers, and digital processing.

Construction



Code for Arduino !

```
// Auduino, the Lo-Fi granular synthesiser
//
// by Peter Knight, Tinker.it http://tinker.it
//
// Help:      http://code.google.com/p/tinkerit/wiki/Auduino
// More help: http://groups.google.com/group/auduino
//
// Analog in 0: Grain 1 pitch
// Analog in 1: Grain 2 decay
// Analog in 2: Grain 1 decay
// Analog in 3: Grain 2 pitch
// Analog in 5: Grain repetition frequency
//
// Digital 3: Audio out (Digital 11 on ATmega8)
//
// Changelog:
// 19 Nov 2008: Added support for ATmega8 boards
// 21 Mar 2009: Added support for ATmega328 boards
```

```
// 7 Apr 2009: Fixed interrupt vector for ATmega328 boards
// 8 Apr 2009: Added support for ATmega1280 boards (Arduino Mega)

#include <avr/io.h>
#include <avr/interrupt.h>

uint16_t syncPhaseAcc;
uint16_t syncPhaseInc;
uint16_t grainPhaseAcc;
uint16_t grainPhaseInc;
uint16_t grainAmp;
uint8_t grainDecay;
uint16_t grain2PhaseAcc;
uint16_t grain2PhaseInc;
uint16_t grain2Amp;
uint8_t grain2Decay;

// Map Analogue channels
#define SYNC_CONTROL         (5)
#define GRAIN_FREQ_CONTROL   (0)
#define GRAIN_DECAY_CONTROL  (2)
#define GRAIN2_FREQ_CONTROL  (3)
#define GRAIN2_DECAY_CONTROL (1)


// Changing these will also requires rewriting audioOn()

#if defined(__AVR_ATmega8__)
//
// On old ATmega8 boards.
//    Output is on pin 11
//
#define LED_PIN       13
#define LED_PORT      PORTB
#define LED_BIT       5
#define PWM_PIN       11
#define PWM_VALUE     OCR2
#define PWM_INTERRUPT TIMER2_OVF_vect
#elif defined(__AVR_ATmega1280__)
//
// On the Arduino Mega
//    Output is on pin 3
//
#define LED_PIN       13
#define LED_PORT      PORTB
#define LED_BIT       7
#define PWM_PIN       3
#define PWM_VALUE     OCR3C
#define PWM_INTERRUPT TIMER3_OVF_vect
```

```
#else
//
// For modern ATmega168 and ATmega328 boards
//    Output is on pin 3
//
#define PWM_PIN       3
#define PWM_VALUE     OCR2B
#define LED_PIN       13
#define LED_PORT      PORTB
#define LED_BIT       5
#define PWM_INTERRUPT TIMER2_OVF_vect
#endif

// Smooth logarithmic mapping
//
uint16_t antilogTable[] = {

64830,64132,63441,62757,62081,61413,60751,60097,59449,58809,58176,57549,56929,56316,55709,
55109,

54515,53928,53347,52773,52204,51642,51085,50535,49991,49452,48920,48393,47871,47356,46846,
46341,

45842,45348,44859,44376,43898,43425,42958,42495,42037,41584,41136,40693,40255,39821,39392,
38968,

38548,38133,37722,37316,36914,36516,36123,35734,35349,34968,34591,34219,33850,33486,33125,
32768
};
uint16_t mapPhaseInc(uint16_t input) {
  return (antilogTable[input & 0x3f]) >> (input >> 6);
}

// Stepped chromatic mapping
//
uint16_t midiTable[] = {
  17,18,19,20,22,23,24,26,27,29,31,32,34,36,38,41,43,46,48,51,54,58,61,65,69,73,
  77,82,86,92,97,103,109,115,122,129,137,145,154,163,173,183,194,206,218,231,
  244,259,274,291,308,326,346,366,388,411,435,461,489,518,549,581,616,652,691,
  732,776,822,871,923,978,1036,1097,1163,1232,1305,1383,1465,1552,1644,1742,
  1845,1955,2071,2195,2325,2463,2610,2765,2930,3104,3288,3484,3691,3910,4143,
  4389,4650,4927,5220,5530,5859,6207,6577,6968,7382,7821,8286,8779,9301,9854,
  10440,11060,11718,12415,13153,13935,14764,15642,16572,17557,18601,19708,20879,
  22121,23436,24830,26306
};
uint16_t mapMidi(uint16_t input) {
  return (midiTable[(1023-input) >> 3]);
}
```

```
// Stepped Pentatonic mapping
//
uint16_t pentatonicTable[54] = {
  0,19,22,26,29,32,38,43,51,58,65,77,86,103,115,129,154,173,206,231,259,308,346,
  411,461,518,616,691,822,923,1036,1232,1383,1644,1845,2071,2463,2765,3288,
  3691,4143,4927,5530,6577,7382,8286,9854,11060,13153,14764,16572,19708,22121,26306
};

uint16_t mapPentatonic(uint16_t input) {
  uint8_t value = (1023-input) / (1024/53);
  return (pentatonicTable[value]);
}


void audioOn() {
#if defined(__AVR_ATmega8__)
  // ATmega8 has different registers
  TCCR2 = _BV(WGM20) | _BV(COM21) | _BV(CS20);
  TIMSK = _BV(TOIE2);
#elif defined(__AVR_ATmega1280__)
  TCCR3A = _BV(COM3C1) | _BV(WGM30);
  TCCR3B = _BV(CS30);
  TIMSK3 = _BV(TOIE3);
#else
  // Set up PWM to 31.25kHz, phase accurate
  TCCR2A = _BV(COM2B1) | _BV(WGM20);
  TCCR2B = _BV(CS20);
  TIMSK2 = _BV(TOIE2);
#endif
}


void setup() {
  pinMode(PWM_PIN,OUTPUT);
  audioOn();
  pinMode(LED_PIN,OUTPUT);
}

void loop() {
  // The loop is pretty simple - it just updates the parameters for the oscillators.
  //
  // Avoid using any functions that make extensive use of interrupts, or turn interrupts off.
  // They will cause clicks and poops in the audio.

  // Smooth frequency mapping
  //syncPhaseInc = mapPhaseInc(analogRead(SYNC_CONTROL)) / 4;

  // Stepped mapping to MIDI notes: C, Db, D, Eb, E, F...
  //syncPhaseInc = mapMidi(analogRead(SYNC_CONTROL));
```

```
  // Stepped pentatonic mapping: D, E, G, A, B
  syncPhaseInc = mapPentatonic(analogRead(SYNC_CONTROL));

  grainPhaseInc  = mapPhaseInc(analogRead(GRAIN_FREQ_CONTROL)) / 2;
  grainDecay     = analogRead(GRAIN_DECAY_CONTROL) / 8;
  grain2PhaseInc = mapPhaseInc(analogRead(GRAIN2_FREQ_CONTROL)) / 2;
  grain2Decay    = analogRead(GRAIN2_DECAY_CONTROL) / 4;
}

SIGNAL(PWM_INTERRUPT)
{
  uint8_t value;
  uint16_t output;

  syncPhaseAcc += syncPhaseInc;
  if (syncPhaseAcc < syncPhaseInc) {
    // Time to start the next grain
    grainPhaseAcc = 0;
    grainAmp = 0x7fff;
    grain2PhaseAcc = 0;
    grain2Amp = 0x7fff;
    LED_PORT ^= 1 << LED_BIT; // Faster than using digitalWrite
  }

  // Increment the phase of the grain oscillators
  grainPhaseAcc += grainPhaseInc;
  grain2PhaseAcc += grain2PhaseInc;

  // Convert phase into a triangle wave
  value = (grainPhaseAcc >> 7) & 0xff;
  if (grainPhaseAcc & 0x8000) value = ~value;
  // Multiply by current grain amplitude to get sample
  output = value * (grainAmp >> 8);

  // Repeat for second grain
  value = (grain2PhaseAcc >> 7) & 0xff;
  if (grain2PhaseAcc & 0x8000) value = ~value;
  output += value * (grain2Amp >> 8);

  // Make the grain amplitudes decay by a factor every sample (exponential decay)
  grainAmp -= (grainAmp >> 8) * grainDecay;
  grain2Amp -= (grain2Amp >> 8) * grain2Decay;

  // Scale output to the available range, clipping if necessary
  output >>= 9;
  if (output > 255) output = 255;

  // Output to PWM (this is faster than using analogWrite)
```

```
  PWM_VALUE = output;
}
```

implementations:

- we can add an LDR (light depending resistor) to the circuitry, instead of a potentiometer, connecting one leg of the LDR to 5v and the other one to A0(or the one you chose) passing through a 10k resistor that goes to GND.

- delay: http://rcarduino.blogspot.de/2012/11/auduino-with-delay.html

- last implementation made by Pechblenda: adding some capacitor sensors and using CapSense Library and code we build Analgital Friction Synth, that uses 3 capacitive sensors to modulate sound with body presence.

http://www.ladyada.net/learn/arduino/lesson1.html

http://findingada.com/about/who-was-ada/

http://findingada.com/book/ada-lovelace-victorian-computing-visionary/

difference between analog / digital
0/255 – 0/1

thanks for all

disfruta // enjoy