Code to program the Arduino

HTTP://POPO.WEBATU.COM

```
/*
============================================================
========                                           =======
========              MIDI XYLOPHONE               =======
========                                           =======
========          Luis Pato   April 2012           =======
========                   V1.0                    =======
========                                           =======
============================================================

Project:
Midi Xylophone, with:
   - 8 Note Pads;
   - 4 Assignable Buttons (with Click and Click+Hold);
   - 4 Assignable Knobs;
   - 3 Selectable Modes (Major, Minor, Pentatonic)

Idea and code based on the Midi Xylophone from Jenna deBoisblanc.
(http://www.jdeboi.com/portfolio/electronics.html)

"ClickButton" library from raronoff@gmail.com
(http://code.google.com/p/clickbutton/)

I need to thank the people from the Arduino.cc forum, specially
Grumpy_Mike for all their help and support.
Without them i would have never been able to finish this project!
*/



#include "ClickButton.h"

// Modes (Major, Minor and Pentatonic)
unsigned char PadNoteMAJOR[8] = {60,62,64,65,67,69,71,72};
unsigned char PadNoteMINOR[8] = {60,62,63,65,67,68,70,72};
unsigned char PadNotePENTA[8] = {55,60,62,64,67,69,72,74};

unsigned char status;
int PadCutOff[8] = {80,110,110,110,80,110,110,110};
int MaxPlayTime[8] = {90,90,90,90,90,90,90,90};
#define midichannel 0;
boolean     activePad[8]        =        {0,0,0,0,0,0,0,0};// Array of flags of pad cu
int      PinPlayTime[8]         =        {0,0,0,0,0,0,0,0};// Counter since pad starte
boolean VelocityFlag  =false;

int  analogPin  =  A0;
int hitavg = 0;
int pad = 0;

byte byte1;
byte byte2;
byte byte3;

int   r0   =   0;//value of select pin at the 4051 (s0)
int   r1   =   0;//value of select pin at the 4051 (s1)
```

```
int   r2   =   0;//value of select pin at the 4051 (s2)
int  count   =  0;//which y pin we are selecting

int multiplex1[8];
int multiplex2[8];
int multiplex3[8];

// For the Buttons
const int buttonPin1 = 8;
const int buttonPin2 = 9;
const int buttonPin3 = 10;
const int buttonPin4 = 11;

ClickButton  button1(buttonPin1,LOW, CLICK_PULLUP);
ClickButton  button2(buttonPin2,LOW, CLICK_PULLUP);
ClickButton  button3(buttonPin3,LOW, CLICK_PULLUP);
ClickButton  button4(buttonPin4,LOW, CLICK_PULLUP);

int lastClickCode1 = 0;
int lastClickCode2 = 0;
int lastClickCode3 = 0;
int lastClickCode4 = 0;

// For the Pots
int potPins[4] = {A1, A2, A3, A4};
int potVal[4];
int potOutput[4];
int potLastOutput[4];

// For the Mode Button
int modeNum = 0;
int modeLed1Pin = 5;
int modeLed2Pin = 6;
int modeLed3Pin = 12;

const int modeButtonPin = 7;
ClickButton  modeButton(modeButtonPin,LOW, CLICK_PULLUP);
int lastClickCodeMode = 0;


//*****************************************************************************
//                                                        Setup
//*****************************************************************************

void setup()
{
  pinMode(2, OUTPUT);      // s0
  pinMode(3, OUTPUT);      // s1
  pinMode(4, OUTPUT);      // s2
  for (int potnumber=0; potnumber<4; potnumber++) {
     potVal[potnumber] =analogRead(potPins[potnumber]);
     potOutput[potnumber] =map(potVal[potnumber], 0, 1023, 0, 127);}
  pinMode(modeButtonPin, INPUT);
  pinMode(modeLed1Pin, OUTPUT);
  pinMode(modeLed2Pin, OUTPUT);
```

```
  pinMode(modeLed3Pin, OUTPUT);
  Serial.begin(57600);
}

//*****************************************************************************************
//                                                    Main Program
//*****************************************************************************************

void loop (){
  midiLoopback();
  readSensors(0);
  //readSensors(1);
  //readSensors(2);
  checkSensors(0);
  //checkSensors(1);
  //checkSensors(2);
  buttons();
  pots();
  mode();
}

// ************************** Mode Selection Button ********************************
void mode() {
 modeButton.Update();
 //Mode 0
if (modeNum == 0) {
   if(modeButton.click  ==CLICK_SINGLECLICKED && modeButton.click != lastClickCodeMode)
   modeNum = 1;}
 else if(modeButton.click  ==CLICK_SINGLEHOLD && modeButton.click != lastClickCodeMode)
   modeNum = 2;}
 lastClickCodeMode = modeButton.click;
 digitalWrite(modeLed1Pin, HIGH);
 digitalWrite(modeLed2Pin, LOW);
 digitalWrite(modeLed3Pin, LOW); }

 //Mode 1
 else if (modeNum == 1) {
      if(modeButton.click  ==CLICK_SINGLECLICKED && modeButton.click != lastClickCodeMo
    modeNum  =  2;}
 else if(modeButton.click  ==CLICK_SINGLEHOLD && modeButton.click != lastClickCodeMode)
    modeNum  =  0;}
   lastClickCodeMode = modeButton.click;
 digitalWrite(modeLed1Pin, LOW);
 digitalWrite(modeLed2Pin, HIGH);
 digitalWrite(modeLed3Pin, LOW); }

  //Mode 2
  else if (modeNum == 2) {
      if(modeButton.click  ==CLICK_SINGLECLICKED && modeButton.click != lastClickCodeMo
    modeNum  =  0;}
 else if(modeButton.click  ==CLICK_SINGLEHOLD && modeButton.click != lastClickCodeMode)
   modeNum  =  1;}
   lastClickCodeMode = modeButton.click;
 digitalWrite(modeLed1Pin, LOW);
 digitalWrite(modeLed2Pin, LOW);
```

```
 digitalWrite(modeLed3Pin, HIGH); }
}

// ******************************** Assignable Buttons ********************************
void buttons() {
  button1.Update();
  button2.Update();
  button3.Update();
  button4.Update();

  // BUTTON 1
  if(button1.click  ==CLICK_SINGLECLICKED && button1.click != lastClickCode1) {
  MIDI_TX(0xB0, 80, 127);}
  if(button1.click  ==CLICK_SINGLEHOLD && button1.click!= lastClickCode1) {
  MIDI_TX(0xB0, 80, 0);}
  lastClickCode1 = button1.click;

  // BUTTON 2
  if(button2.click  ==CLICK_SINGLECLICKED && button2.click != lastClickCode2) {
  MIDI_TX(0xB0, 81, 127);}
  if(button2.click  ==CLICK_SINGLEHOLD && button2.click!= lastClickCode2) {
  MIDI_TX(0xB0, 81, 0);}
  lastClickCode2 = button2.click;

  // BUTTON 3
  if(button3.click  ==CLICK_SINGLECLICKED && button3.click != lastClickCode3) {
  MIDI_TX(0xB0, 82, 127);}
  if(button3.click  ==CLICK_SINGLEHOLD && button3.click!= lastClickCode3) {
  MIDI_TX(0xB0, 82, 0);}
  lastClickCode3 = button3.click;

  // BUTTON 4
  if(button4.click  ==CLICK_SINGLECLICKED && button4.click != lastClickCode4) {
  MIDI_TX(0xB0, 83, 127);}
  if(button4.click  ==CLICK_SINGLEHOLD && button4.click!= lastClickCode4) {
  MIDI_TX(0xB0, 83, 0);}
  lastClickCode4 = button4.click;
}

// ******************************** Assignable Knobs ********************************
void pots() {
  for (int potnumber=0; potnumber<4; potnumber++) {
     potVal[potnumber] =analogRead(potPins[potnumber]);
     potOutput[potnumber] =map(potVal[potnumber], 0, 1023, 0, 127);
    if (potOutput[potnumber] != potLastOutput[potnumber]) {
       MIDI_TX(0xB0, 7+potnumber, potOutput[potnumber]);//Control Change (Ch 1), Contro
      potLastOutput[potnumber] = potOutput[potnumber];
    }
  }
}

// ********************************** Read Sensors **********************************
void readSensors (int analogPin) {
  for (count=0; count<=7; count++) {
```

```
    // select the bit
     r2 =bitRead(count,0);    // use this with arduino 0013 (and newer versions)
     r1 =bitRead(count,1);    // use this with arduino 0013 (and newer versions)
     r0 =bitRead(count,2);    // use this with arduino 0013 (and newer versions)

    //r0 = count & 0x01;      // old version of setting the bits
    //r1 = (count>>1) & 0x01;      // old version of setting the bits
    //r2 = (count>>2) & 0x01;

    digitalWrite(2, r0);
    digitalWrite(3, r1);
    digitalWrite(4, r2);

      //Read and store the input value at a location in the array
      //if(analogPin==0){

      multiplex1[count] =analogRead(analogPin);
    //Serial.print(count);
     //Serial.print(" ");
     //Serial.println(multiplex1[count]);
      // }
      //else if(analogPin==1){
      //  multiplex2[count] = analogRead(analogPin);
      //
      //else if(analogPin==2){
      //  multiplex3[count] = analogRead(analogPin);
      // }
    }
}

// ************************ Playing the notes (from right Mode) ************************
void checkSensors(int analogPin){

  // Major Scale
  if (modeNum == 0) {
    for(int pin=0; pin <=7; pin++){
      if(analogPin==0){
        hitavg = multiplex1[pin];
        pad=pin;
      }
      else if(analogPin==1){
        hitavg = multiplex2[pin];
        pad=pin+8;
      }
      else if(analogPin==2){
        hitavg = multiplex3[pin];
        pad=pin+16;
      }

      if((hitavg > PadCutOff[pin])){
        if((activePad[pad]  ==false)){
          if(VelocityFlag  ==true){
            hitavg = (hitavg / 8) -1;
          }
          else{
```

```
            hitavg = 127;
          }
          MIDI_TX(144,PadNoteMAJOR[pad],hitavg);
          PinPlayTime[pad] = 0;
           activePad[pad] =true;
        }
        else {
          PinPlayTime[pad] = PinPlayTime[pad] + 1;
        }
      }
      else if((activePad[pad]  ==true)) //the pad is active, but it is not greater than
      {
        PinPlayTime[pad] = PinPlayTime[pad] + 1;
        if(PinPlayTime[pad] > MaxPlayTime[pad]) {
           activePad[pad] =false;
          MIDI_TX(128,PadNoteMAJOR[pad],127);
        }
      }
    }
  }

// Minor Scale
if(modeNum == 1) {
  for(int pin=0; pin <=7; pin++){
    if(analogPin==0){
      hitavg = multiplex1[pin];
      pad=pin;
    }
    else if(analogPin==1){
      hitavg = multiplex2[pin];
      pad=pin+8;
    }
    else if(analogPin==2){
      hitavg = multiplex3[pin];
      pad=pin+16;
    }

    if((hitavg > PadCutOff[pin])){
      if((activePad[pad]  ==false)){
        if(VelocityFlag  ==true){
          hitavg = (hitavg / 8) -1;
        }
        else{
          hitavg = 127;
        }
        MIDI_TX(144,PadNoteMINOR[pad],hitavg);
        PinPlayTime[pad] = 0;
         activePad[pad] =true;
      }
      else {
        PinPlayTime[pad] = PinPlayTime[pad] + 1;
      }
    }
    else if((activePad[pad]  ==true)) //the pad is active, but it is not greater than cu
    {
```

```
      PinPlayTime[pad] = PinPlayTime[pad] + 1;
      if(PinPlayTime[pad] > MaxPlayTime[pad])
      {
         activePad[pad] =false;
        MIDI_TX(128,PadNoteMINOR[pad],127);
      }
    }
  }
}

// Pentatonic Scale
if (modeNum == 2) {
  for(int pin=0; pin <=7; pin++){
    if(analogPin==0){
      hitavg = multiplex1[pin];
      pad=pin;
    }
    else if(analogPin==1){
      hitavg = multiplex2[pin];
      pad=pin+8;
    }
    else if(analogPin==2){
      hitavg = multiplex3[pin];
      pad=pin+16;
    }
    if((hitavg > PadCutOff[pin])){
      if((activePad[pad]  ==false)){
        if(VelocityFlag  ==true){
          hitavg = (hitavg / 8) -1;
        }
        else{
          hitavg = 127;
        }
        MIDI_TX(144,PadNotePENTA[pad],hitavg);
        PinPlayTime[pad] = 0;
         activePad[pad] =true;
      }
      else {
        PinPlayTime[pad] = PinPlayTime[pad] + 1;
      }
    }
    else if((activePad[pad]  ==true)) //the pad is active, but it is not greater than c
    {
      PinPlayTime[pad] = PinPlayTime[pad] + 1;
      if(PinPlayTime[pad] > MaxPlayTime[pad])
      {
         activePad[pad] =false;
        MIDI_TX(128,PadNotePENTA[pad],127);
      }
    }
  }
}
}

// ******************************************** MIDI LOOPBACK ***************************
```

```
void midiLoopback(){
  if(Serial.available() > 0){
       byte1 =Serial.read();
       byte2 =Serial.read();
       byte3 =Serial.read();
      MIDI_TX(byte1, byte2, byte3);
    }
  }

//*********************************************************************************
//                                          Transmit MIDI Message
//*********************************************************************************
void MIDI_TX(unsigned char  MESSAGE,unsigned char  PITCH,unsigned char VELOCITY)
{
  status = MESSAGE + midichannel;
  Serial.write(status);
  Serial.write(PITCH);
  Serial.write(VELOCITY);
}

//*********************************************************************************
// Version v1.0 (April 2012)
// Prototype I finnished and tested on 17th April 2012
// =)
```