// Pete McBennett <https://www.youtube.com/channel/UCk4mtz-tZbXdk1Xb0DSd2QQ/videos>

// Numerical values that the digital oscillator references to produce the various notes

float C1 = 40;

float Db1 = 42;

float D1 = 45;

float Eb1 = 48;

float E1 = 50;

float F1 = 53;

float Gb1 = 56;

float G1 = 60;

float Ab1 = 63;

float A\_1 = 67;

float Bb1 = 71;

float B\_1 = 75;

float C2 = 79;

float Db2 = 83;

float D2 = 88;

float Eb2 = 93;

float E2 = 98;

float F2 = 103;

float Gb2 = 109;

float G2 = 115;

float Ab2 = 121;

float A\_2 = 128;

float Bb2 = 135;

float B2 = 142;

float C3 = 149;

float Db3 = 157;

float D3 = 165;

float Eb3 = 173;

float E3 = 182;

float F3 = 191;

// Note pedal names

float key1;

float key2;

float key3;

float key4;

float key5;

// Variables for PWM

byte wave;

volatile long t;

float period;

volatile float pulseWidthScaled = 0;

volatile float pulseWidth = 0;

volatile float pulseWidthMin = 0;

volatile byte direct = 0;

volatile float rate = 0;

volatile float pulseK = 0;

int trg = 1;

float freq = 0;

int buttonCount = 1; //counter for the number of transpose pedal pushes

int buttonState = 0; //current state of the pedal

int lastButtonState = 0; //previous state of the pedal

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

void setup() {

pinMode(A1,INPUT\_PULLUP); //pedal inputs for five notes

pinMode(A2,INPUT\_PULLUP);

pinMode(A3,INPUT\_PULLUP);

pinMode(A4,INPUT\_PULLUP);

pinMode(A5,INPUT\_PULLUP);

pinMode(5,INPUT\_PULLUP); //pedal input for mute

pinMode(6,INPUT\_PULLUP); //pedal input for transpose

pinMode(13,OUTPUT); //audio output

pinMode(7,OUTPUT); //trigger output

delay(50);

//---------------------------------------TIMER / INTERRUPT SETUP--------------------------------------

//cli(); //disable interrupts

//timer 1:

TCCR1B = 0; //set entire TCCR1B register to 0

TCCR1B |= (1 << WGM12); //turn on CTC mode

TCCR1B |= (1 << CS10); // Set CS10 bit for 0 prescaler

TIMSK1 |= (1 << OCIE1A); // enable timer compare interrupt

//sei(); //enable interrupts

}

// END SETUP /////////////////////////////////////////////////

//------------------------------------INTERRUPT SERVICE ROUTINE FOR AUDIO OUTPUT WITH PWM------------------------------------------

ISR(TIMER1\_COMPA\_vect){

if (t < pulseWidthScaled) {

wave = B111111; //output pin 13 high on PORTB

}

else{

wave = B011111; //output pin 13 low on PORTB

}

t +=1;

if (t >= period){ //check period and adjust pulse width

t = 0;

if(direct == 0){

if(pulseWidthScaled <= pulseWidthMin){

direct = 1;

pulseWidthScaled = pulseWidthMin + rate;

}

else{

pulseWidthScaled = pulseWidthScaled - rate;

}

}

if(direct == 1){

if(pulseWidthScaled >= pulseWidth){

direct = 0;

pulseWidthScaled = pulseWidth - rate;

}

else{

pulseWidthScaled = pulseWidthScaled + rate;

}

}

}

PORTB = wave; //PORTB manipulation to produce audio output

}

//------------------------------------START LOOP-------------------------------------------------------

void loop() {

rate = (pulseK\*period/40000); // sets PWM speed, 40000 sounds nice

buttonState = digitalRead(6); // read the transpose pedal input pin

if (buttonState != lastButtonState) // compare the buttonState to its previous state

if (buttonState == LOW) // if the state has changed, and the current state is HIGH, then increment the counter:

buttonCount++;

lastButtonState = buttonState; // save the current state as the last state for next time through the loop

// The following five groups of five keys (pedals) can be changed to any

// notes desired from the list at the beginning of the code.

// This default selection produces five useful pentatonic scales

// that can be cycled through using the transpose pedal.

if(buttonCount == 1) //root note E1

{

key1 = E1;

key2 = G1;

key3 = A\_1;

key4 = B\_1;

key5 = D2;

}

if(buttonCount == 2) //root note G1

{

key1 = G1;

key2 = Bb1;

key3 = C2;

key4 = D2;

key5 = F2;

}

if(buttonCount == 3) //root note A1

{

key1 = A\_1;

key2 = C2;

key3 = D2;

key4 = E2;

key5 = G2;

}

if(buttonCount == 4) //root note B1

{

key1 = B\_1;

key2 = D2;

key3 = E2;

key4 = Gb2;

key5 = A\_2;

}

if(buttonCount == 5) //root note D2

{

key1 = D2;

key2 = F2;

key3 = G2;

key4 = A\_2;

key5 = C3;

}

if(buttonCount == 6) //return to root note E1

buttonCount = 1;

if(digitalRead(5)==LOW) //mute pedal

{

freq = 0;

period = 0;

pulseWidth = period \* 0.5;

pulseWidthMin = period \* 0.2;

pulseWidthScaled = 0;

pulseK = pulseWidth - pulseWidthMin;

}

else if(digitalRead(A5)==LOW) //pedal 5

{

if(freq != key5)

trg = 1;

if(trg == 1)

digitalWrite(7, HIGH);

digitalWrite(7, LOW);

trg = 0;

freq = key5;

period = 50000/freq;

pulseWidth = period \* 0.5;

pulseWidthMin = period \* 0.2;

pulseK = pulseWidth - pulseWidthMin;

}

else if(digitalRead(A4)==LOW) //pedal 4

{

if(freq != key4)

trg = 1;

if(trg == 1)

digitalWrite(7, HIGH);

digitalWrite(7, LOW);

trg = 0;

freq = key4;

period = 50000/freq;

pulseWidth = period \* 0.5;

pulseWidthMin = period \* 0.2;

pulseK = pulseWidth - pulseWidthMin;

}

else if(digitalRead(A3)==LOW) //pedal 3

{

if(freq != key3)

trg = 1;

if(trg == 1)

digitalWrite(7, HIGH);

digitalWrite(7, LOW);

trg = 0;

freq = key3;

period = 50000/freq;

pulseWidth = period \* 0.5;

pulseWidthMin = period \* 0.2;

pulseK = pulseWidth - pulseWidthMin;

}

else if(digitalRead(A2)==LOW) //pedal 2

{

if(freq != key2)

trg = 1;

if(trg == 1)

digitalWrite(7, HIGH);

digitalWrite(7, LOW);

trg = 0;

freq = key2;

period = 50000/freq;

pulseWidth = period \* 0.5;

pulseWidthMin = period \* 0.2;

pulseK = pulseWidth - pulseWidthMin;

}

else if(digitalRead(A1)==LOW) //pedal 1

{

if(freq != key1)

trg = 1;

if(trg == 1)

digitalWrite(7, HIGH);

digitalWrite(7, LOW);

trg = 0;

freq = key1;

period = 50000/freq;

pulseWidth = period \* 0.5;

pulseWidthMin = period \* 0.2;

pulseK = pulseWidth - pulseWidthMin;

}

else //reset trigger

{

trg = 1;

}

}