/\*

 Arduino Mario Bros Tunes Plus LED's

 outside run around LED's on pin 2,3,4,5,6,7

 Mario and Luigi eyes to pin 13

 With Piezo Buzzer and PWM

 //Modified by Pat McMahon 4/6/2022 Connect piezo + to D3, piezo - to Gnd. Sounds OK without resistor.

 //A060- Mario Bros Tune

 Connect the positive side of the Buzzer to pin 3,

 then the negative side to a 1k ohm resistor. Connect

 the other side of the 1 k ohm resistor to

 ground(GND) pin on the Arduino.

 by: Dipto Pratyaksa

 last updated: 31/3/13

\*/

 int delayTime1=70; //milliseconds in between LED's.

 int runXtimes=30; //Times it runs through the loop

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* Public Constants

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define NOTE\_B0 31

#define NOTE\_C1 33

#define NOTE\_CS1 35

#define NOTE\_D1 37

#define NOTE\_DS1 39

#define NOTE\_E1 41

#define NOTE\_F1 44

#define NOTE\_FS1 46

#define NOTE\_G1 49

#define NOTE\_GS1 52

#define NOTE\_A1 55

#define NOTE\_AS1 58

#define NOTE\_B1 62

#define NOTE\_C2 65

#define NOTE\_CS2 69

#define NOTE\_D2 73

#define NOTE\_DS2 78

#define NOTE\_E2 82

#define NOTE\_F2 87

#define NOTE\_FS2 93

#define NOTE\_G2 98

#define NOTE\_GS2 104

#define NOTE\_A2 110

#define NOTE\_AS2 117

#define NOTE\_B2 123

#define NOTE\_C3 131

#define NOTE\_CS3 139

#define NOTE\_D3 147

#define NOTE\_DS3 156

#define NOTE\_E3 165

#define NOTE\_F3 175

#define NOTE\_FS3 185

#define NOTE\_G3 196

#define NOTE\_GS3 208

#define NOTE\_A3 220

#define NOTE\_AS3 233

#define NOTE\_B3 247

#define NOTE\_C4 262

#define NOTE\_CS4 277

#define NOTE\_D4 294

#define NOTE\_DS4 311

#define NOTE\_E4 330

#define NOTE\_F4 349

#define NOTE\_FS4 370

#define NOTE\_G4 392

#define NOTE\_GS4 415

#define NOTE\_A4 440

#define NOTE\_AS4 466

#define NOTE\_B4 494

#define NOTE\_C5 523

#define NOTE\_CS5 554

#define NOTE\_D5 587

#define NOTE\_DS5 622

#define NOTE\_E5 659

#define NOTE\_F5 698

#define NOTE\_FS5 740

#define NOTE\_G5 784

#define NOTE\_GS5 831

#define NOTE\_A5 880

#define NOTE\_AS5 932

#define NOTE\_B5 988

#define NOTE\_C6 1047

#define NOTE\_CS6 1109

#define NOTE\_D6 1175

#define NOTE\_DS6 1245

#define NOTE\_E6 1319

#define NOTE\_F6 1397

#define NOTE\_FS6 1480

#define NOTE\_G6 1568

#define NOTE\_GS6 1661

#define NOTE\_A6 1760

#define NOTE\_AS6 1865

#define NOTE\_B6 1976

#define NOTE\_C7 2093

#define NOTE\_CS7 2217

#define NOTE\_D7 2349

#define NOTE\_DS7 2489

#define NOTE\_E7 2637

#define NOTE\_F7 2794

#define NOTE\_FS7 2960

#define NOTE\_G7 3136

#define NOTE\_GS7 3322

#define NOTE\_A7 3520

#define NOTE\_AS7 3729

#define NOTE\_B7 3951

#define NOTE\_C8 4186

#define NOTE\_CS8 4435

#define NOTE\_D8 4699

#define NOTE\_DS8 4978

#define melodyPin 9

//Mario main theme melody

int melody[] = {

 NOTE\_E7, NOTE\_E7, 0, NOTE\_E7,

 0, NOTE\_C7, NOTE\_E7, 0,

 NOTE\_G7, 0, 0, 0,

 NOTE\_G6, 0, 0, 0,

 NOTE\_C7, 0, 0, NOTE\_G6,

 0, 0, NOTE\_E6, 0,

 0, NOTE\_A6, 0, NOTE\_B6,

 0, NOTE\_AS6, NOTE\_A6, 0,

 NOTE\_G6, NOTE\_E7, NOTE\_G7,

 NOTE\_A7, 0, NOTE\_F7, NOTE\_G7,

 0, NOTE\_E7, 0, NOTE\_C7,

 NOTE\_D7, NOTE\_B6, 0, 0,

 NOTE\_C7, 0, 0, NOTE\_G6,

 0, 0, NOTE\_E6, 0,

 0, NOTE\_A6, 0, NOTE\_B6,

 0, NOTE\_AS6, NOTE\_A6, 0,

 NOTE\_G6, NOTE\_E7, NOTE\_G7,

 NOTE\_A7, 0, NOTE\_F7, NOTE\_G7,

 0, NOTE\_E7, 0, NOTE\_C7,

 NOTE\_D7, NOTE\_B6, 0, 0

};

//Mario main them tempo

int tempo[] = {

 12, 12, 12, 12,

 12, 12, 12, 12,

 12, 12, 12, 12,

 12, 12, 12, 12,

 12, 12, 12, 12,

 12, 12, 12, 12,

 12, 12, 12, 12,

 12, 12, 12, 12,

 9, 9, 9,

 12, 12, 12, 12,

 12, 12, 12, 12,

 12, 12, 12, 12,

 12, 12, 12, 12,

 12, 12, 12, 12,

 12, 12, 12, 12,

 12, 12, 12, 12,

 9, 9, 9,

 12, 12, 12, 12,

 12, 12, 12, 12,

 12, 12, 12, 12,

};

//Underworld melody

int underworld\_melody[] = {

 NOTE\_C4, NOTE\_C5, NOTE\_A3, NOTE\_A4,

 NOTE\_AS3, NOTE\_AS4, 0,

 0,

 NOTE\_C4, NOTE\_C5, NOTE\_A3, NOTE\_A4,

 NOTE\_AS3, NOTE\_AS4, 0,

 0,

 NOTE\_F3, NOTE\_F4, NOTE\_D3, NOTE\_D4,

 NOTE\_DS3, NOTE\_DS4, 0,

 0,

 NOTE\_F3, NOTE\_F4, NOTE\_D3, NOTE\_D4,

 NOTE\_DS3, NOTE\_DS4, 0,

 0, NOTE\_DS4, NOTE\_CS4, NOTE\_D4,

 NOTE\_CS4, NOTE\_DS4,

 NOTE\_DS4, NOTE\_GS3,

 NOTE\_G3, NOTE\_CS4,

 NOTE\_C4, NOTE\_FS4, NOTE\_F4, NOTE\_E3, NOTE\_AS4, NOTE\_A4,

 NOTE\_GS4, NOTE\_DS4, NOTE\_B3,

 NOTE\_AS3, NOTE\_A3, NOTE\_GS3,

 0, 0, 0

};

//Underwolrd tempo

int underworld\_tempo[] = {

 12, 12, 12, 12,

 12, 12, 6,

 3,

 12, 12, 12, 12,

 12, 12, 6,

 3,

 12, 12, 12, 12,

 12, 12, 6,

 3,

 12, 12, 12, 12,

 12, 12, 6,

 6, 18, 18, 18,

 6, 6,

 6, 6,

 6, 6,

 18, 18, 18, 18, 18, 18,

 10, 10, 10,

 10, 10, 10,

 3, 3, 3

};

void setup(void)

{

 pinMode(9,OUTPUT);//buzzer

 pinMode(13,OUTPUT);//led indicator when singing a note

 pinMode(2,OUTPUT);//Blue LED

 pinMode(3,OUTPUT);//White LED

 pinMode(4,OUTPUT);//Green LED

 pinMode(5,OUTPUT);//Orange LED

 pinMode(6,OUTPUT);//Yellow LED

 pinMode(7,OUTPUT);//Red LED

}

void loop()

{

 if(runXtimes)

 {

digitalWrite(2,HIGH);delay(delayTime1);digitalWrite(2,LOW);delay(delayTime1);//LED2 on then off

digitalWrite(3,HIGH);delay(delayTime1);digitalWrite(3,LOW);delay(delayTime1);//LED3 on then off

digitalWrite(4,HIGH);delay(delayTime1);digitalWrite(4,LOW);delay(delayTime1);//LED4 on then off

digitalWrite(5,HIGH);delay(delayTime1);digitalWrite(5,LOW);delay(delayTime1);//LED5 on then off

digitalWrite(6,HIGH);delay(delayTime1);digitalWrite(6,LOW);delay(delayTime1);//LED6 on then off

digitalWrite(7,HIGH);delay(delayTime1);digitalWrite(7,LOW);delay(delayTime1);//LED7 on then off

 runXtimes--;

 }

 //sing the tunes

 sing(1);

 sing(1);

 sing(2);

}

int song = 0;

void sing(int s) {

 // iterate over the notes of the melody:

 song = s;

 if (song == 2) {

 Serial.println(" 'Underworld Theme'");

 int size = sizeof(underworld\_melody) / sizeof(int);

 for (int thisNote = 0; thisNote < size; thisNote++) {

 // to calculate the note duration, take one second

 // divided by the note type.

 //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.

 int noteDuration = 1000 / underworld\_tempo[thisNote];

 buzz(melodyPin, underworld\_melody[thisNote], noteDuration);

 // to distinguish the notes, set a minimum time between them.

 // the note's duration + 30% seems to work well:

 int pauseBetweenNotes = noteDuration \* 1.30;

 delay(pauseBetweenNotes);

 // stop the tone playing:

 buzz(melodyPin, 0, noteDuration);

 }

 } else {

 if(runXtimes)

 {

digitalWrite(7,HIGH);delay(delayTime1);digitalWrite(7,LOW);delay(delayTime1);//LED7 on then off

digitalWrite(6,HIGH);delay(delayTime1);digitalWrite(6,LOW);delay(delayTime1);//LED6 on then off

digitalWrite(5,HIGH);delay(delayTime1);digitalWrite(5,LOW);delay(delayTime1);//LED5 on then off

digitalWrite(4,HIGH);delay(delayTime1);digitalWrite(4,LOW);delay(delayTime1);//LED4 on then off

digitalWrite(3,HIGH);delay(delayTime1);digitalWrite(3,LOW);delay(delayTime1);//LED3 on then off

digitalWrite(2,HIGH);delay(delayTime1);digitalWrite(2,LOW);delay(delayTime1);//LED2 on then off

 runXtimes--;

 }

 Serial.println(" 'Mario Theme'");

 int size = sizeof(melody) / sizeof(int);

 for (int thisNote = 0; thisNote < size; thisNote++) {

 // to calculate the note duration, take one second

 // divided by the note type.

 //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.

 int noteDuration = 1000 / tempo[thisNote];

 buzz(melodyPin, melody[thisNote], noteDuration);

 // to distinguish the notes, set a minimum time between them.

 // the note's duration + 30% seems to work well:

 int pauseBetweenNotes = noteDuration \* 1.30;

 delay(pauseBetweenNotes);

 // stop the tone playing:

 buzz(melodyPin, 0, noteDuration);

 }

 }

}

void buzz(int targetPin, long frequency, long length) {

 digitalWrite(13, HIGH);

 long delayValue = 1000000 / frequency / 2; // calculate the delay value between transitions

 //// 1 second's worth of microseconds, divided by the frequency, then split in half since

 //// there are two phases to each cycle

 long numCycles = frequency \* length / 1000; // calculate the number of cycles for proper timing

 //// multiply frequency, which is really cycles per second, by the number of seconds to

 //// get the total number of cycles to produce

 for (long i = 0; i < numCycles; i++) { // for the calculated length of time...

 digitalWrite(targetPin, HIGH); // write the buzzer pin high to push out the diaphram

 delayMicroseconds(delayValue); // wait for the calculated delay value

 digitalWrite(targetPin, LOW); // write the buzzer pin low to pull back the diaphram

 delayMicroseconds(delayValue); // wait again or the calculated delay value

 }

 digitalWrite(13, LOW);

}