```
//   JONAS RUNA:  < CALL CENTER >

//  1. pitched bells

#include <Metro.h>
#include <SoftwareSerial.h>
#include <DFPlayer_Mini_Mp3.h>
#include <wavTrigger.h>
#include <AltSoftSerial.h>

int myList[32] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 19,
                  20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32
                 };

// FUNCOES do WAV TRIGGER

wavTrigger wTrig;

void wavnormal() {
  digitalWrite(2, HIGH);
  delay(10);
  digitalWrite(2, LOW);
}
void wavstop() {
  digitalWrite(3, HIGH);
  delay(10);
  digitalWrite(3, LOW);
}
void wavnext() {
  digitalWrite(4, HIGH);
  delay(10);
  digitalWrite(4, LOW);
}
void wavprev() {
  digitalWrite(6, HIGH);
  delay(10);
  digitalWrite(6, LOW);
}

void walkforward(int numb) {
  for (int j = 0; j < numb; j++) {
    wavstop();
    delay(100);
    wavnext();
    delay(100);
    wavstop();
    delay(100);
```

```
  };
}

// 0. DFPlayer-mini mp3 module

void playsnd1 (int nbr, int dur) {
  mp3_set_serial (Serial); mp3_play (nbr); delay (dur); mp3_stop ();
}

void playsnd2 (int nbr, int dur) {
  mp3_set_serial (Serial1); mp3_play (nbr); delay (dur); mp3_stop ();
}

void playsnd3 (int nbr, int dur) {
  mp3_set_serial (Serial2); mp3_play (nbr); delay (dur); mp3_stop ();
}

void playsnd4 (int nbr, int dur) {
  mp3_set_serial (Serial3); mp3_play (nbr); delay (dur); mp3_stop ();
}

void playsnd1234 (int nbr1, int nbr2, int nbr3, int nbr4, int dur) {
  if (nbr1 > 0) {
    mp3_set_serial (Serial); mp3_play (nbr1);
  }
  if (nbr2 > 0) {
    mp3_set_serial (Serial1); mp3_play (nbr2);
  }
  if (nbr3 > 0) {
    mp3_set_serial (Serial2); mp3_play (nbr3);
  }
  if (nbr4 > 0) {
    mp3_set_serial (Serial3); mp3_play (nbr4);
  }
  delay (dur);
  if (nbr4 > 0) {
    mp3_set_serial (Serial3); mp3_stop ();
  }
  if (nbr3 > 0) {
    mp3_set_serial (Serial2); mp3_stop ();
  }
  if (nbr2 > 0) {
    mp3_set_serial (Serial1); mp3_stop ();
  }
  if (nbr1 > 0) {
    mp3_set_serial (Serial); mp3_stop ();
  }
}
```

```
void startsnd4(int nbr1, int nbr2, int nbr3, int nbr4) {
  if (nbr1 > 0) {
    mp3_set_serial (Serial); mp3_play (nbr1);
  }
  if (nbr2 > 0) {
    mp3_set_serial (Serial1); mp3_play (nbr2);
  }
  if (nbr3 > 0) {
    mp3_set_serial (Serial2); mp3_play (nbr3);
  }
  if (nbr4 > 0) {
    mp3_set_serial (Serial3); mp3_play (nbr4);
  }
}

void startanysnd4() {
  int mp3a[] = { random(1,74), random(102,107), random(200,276),
random(300,305), random(401,485), random(500,502) };
  int mp3b[] = { random(1,74), random(102,107), random(200,276),
random(300,305), random(401,485), random(500,502) };
  int mp3c[] = { random(1,74), random(102,107), random(200,276),
random(300,305), random(401,485), random(500,502) };
  int mp3d[] = { random(1,74), random(102,107), random(200,276),
random(300,305), random(401,485), random(500,502) };
  startsnd4(mp3a[random(0,6)], mp3b[random(0,6)], mp3c[random(0,6)],
mp3d[random(0,6)]);
}

void stopsnd4(int nbr1, int nbr2, int nbr3, int nbr4) {
  if (nbr4 > 0) {
    mp3_set_serial (Serial3); mp3_stop ();
  }
  if (nbr3 > 0) {
    mp3_set_serial (Serial2); mp3_stop ();
  }
  if (nbr2 > 0) {
    mp3_set_serial (Serial1); mp3_stop ();
  }
  if (nbr1 > 0) {
    mp3_set_serial (Serial); mp3_stop ();
  }
}


void onsetsnd4(int nbr1, int nbr2, int nbr3, int nbr4, int on1, int on2, int
on3, int on4) {
  delay(on1);
```

```
  if (nbr1 > 0) {
    mp3_set_serial (Serial); mp3_play (nbr1);
  }
  delay(on2);
  if (nbr2 > 0) {
    mp3_set_serial (Serial1); mp3_play (nbr2);
  }
  delay(on3);
  if (nbr3 > 0) {
    mp3_set_serial (Serial2); mp3_play (nbr3);
  }
  delay(on4);
  if (nbr4 > 0) {
    mp3_set_serial (Serial3); mp3_play (nbr4);
  }
}

// 1. shuffle an array

void bubbleUnsort(int *list, int elem)
{
  for (int a = elem - 1; a > 0; a--)
  {
    int r = random(a + 1);
    if (r != a)
    {
      int temp = list[a];
      list[a] = list[r];
      list[r] = temp;
    }
  }
}

// 2. makenote

int makenote2( int duration, int bellnbr) {
  digitalWrite(bellnbr + 21, HIGH);
  delay(duration);
  digitalWrite(bellnbr + 21, LOW);
}

// 3. makechord

int makechord(int duration, int bellsize, int mybells[]) {
  for (int j = 0; j < bellsize; j++) {
    digitalWrite(mybells[j] + 21, HIGH);
  };
  delay(duration);
```

```
  for (int j = 0; j < bellsize; j++) {
    digitalWrite(mybells[j] + 21, LOW);
  };
}

// 4. makenotedyn

int makenotedyn(int duration, int dyn) {
  int seed = random(1, 10000);
  int myList[32];
  for (int x = 0; x < 32; x++) {
    myList[x] = x + 1;
  };
  bubbleUnsort(myList, sizeof(myList) / sizeof(int));
  int myList2[dyn];
  for (int x = 0; x < dyn; x++) {
    myList2[x] = myList[x];
  };
  makechord(duration, (sizeof(myList2) / sizeof(int)), myList2);
}

void notedyn(float duration, int dyn, int noterestpercentage) {
  int seed = random(1, 10000);
  int soundp = round(duration * (noterestpercentage / 100.0));
  bubbleUnsort(myList, 32);
  makechord(soundp, dyn, myList);
  delay((duration - soundp));
}

// 5. Cluster (type 1 = up, type 2 = down, type 3 = up/down)



int cluster(int totd, int type, int dyn) {
  int seed = random(1, 10000);
  int myList[32] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 19,
                    20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32
                   };
  bubbleUnsort(myList, 32);
  int myList2[dyn];
  for (int x = 0; x < dyn; x++) {
    myList2[x] = myList[x];
  };

  switch (type) {
    case 1: {
        int plen = (totd / (dyn));
```

```
        for (int x = 0; x < dyn; x++) {
          digitalWrite(myList2[x] + 21, HIGH);
          delay(plen);
        };
        for (int x = 0; x < dyn; x++) {
          digitalWrite(myList2[x] + 21, LOW);
        };
      }; break;
    case 2: {
        int plen = (totd / (dyn));
        for (int x = 0; x < dyn; x++) {
          digitalWrite(myList2[x] + 21, HIGH);
        };
        for (int x = 0; x < dyn; x++) {
          delay(plen);
          digitalWrite(myList2[x] + 21, LOW);
        };
      }; break;
    case 3: {
        int plen = (totd / (1 + (dyn * 2)));
        for (int x = 0; x < dyn; x++) {
          // Serial.println(myList2[x]);
          digitalWrite(myList2[x] + 21, HIGH);
          delay(plen);
        };
        delay(plen);
        for (int x = 0; x < dyn; x++) {
          // Serial.println(myList2[dyn - 1 - x]);
          delay(plen);
          digitalWrite(myList2[dyn - 1 - x] + 21, LOW);
        };
      }; break;
  }
}

int cluster2(int totd, int type, int dyn) {
  int seed = random(1, 10000);
  int plen;
  int remainingdur;
  bubbleUnsort(myList, 32);
  if (type < 3) {
    plen = (totd / (dyn));
    remainingdur = (totd - (dyn * plen));
  }
  else {
    plen = (totd / (1 + (dyn * 2)));
    remainingdur = (totd - (plen * (1 + (dyn * 2))));
  };
```

```
  switch (type) {
    case 1: {
        for (int x = 0; x < dyn; x++) {
          digitalWrite(myList[x] + 21, HIGH);
          delay(plen);
        };
        for (int x = 0; x < dyn; x++) {
          digitalWrite(myList[x] + 21, LOW);
        };
      }; break;
    case 2: {
        for (int x = 0; x < dyn; x++) {
          digitalWrite(myList[x] + 21, HIGH);
        };
        for (int x = 0; x < dyn; x++) {
          delay(plen);
          digitalWrite(myList[x] + 21, LOW);
        };
      }; break;
    case 3: {
        for (int x = 0; x < dyn; x++) {
          digitalWrite(myList[x] + 21, HIGH);
          delay(plen);
        };
        delay(plen);
        for (int x = 0; x < dyn; x++) {
          delay(plen);
          digitalWrite(myList[dyn - 1 - x] + 21, LOW);
        };
      }; break;
  }
  if (remainingdur > 3 ) {
    delay((remainingdur - 3));
  };
}

int cluster2wt(int totd, int type, int dyn) {
  int seed = random(1, 10000);
  int plen;
  int remainingdur;
  bubbleUnsort(myList, 32);
  if (type < 3) {
    plen = (totd / (dyn));
    remainingdur = (totd - (dyn * plen));
  }
  else {
    plen = (totd / (1 + (dyn * 2)));
    remainingdur = (totd - (plen * (1 + (dyn * 2))));
```

```
  };
  switch (type) {
    case 1: {
        for (int x = 0; x < dyn; x++) {
          digitalWrite(myList[x] + 21, HIGH);
          digitalWrite(4, HIGH);
          delay(plen);
        };
        for (int x = 0; x < dyn; x++) {
          digitalWrite(myList[x] + 21, LOW);
          digitalWrite(4, LOW);
        };
      }; break;
    case 2: {
        for (int x = 0; x < dyn; x++) {
          digitalWrite(myList[x] + 21, HIGH);
          digitalWrite(4, HIGH);
        };
        for (int x = 0; x < dyn; x++) {
          delay(plen);
          digitalWrite(myList[x] + 21, LOW);
          digitalWrite(4, LOW);
        };
      }; break;
    case 3: {
        for (int x = 0; x < dyn; x++) {
          digitalWrite(myList[x] + 21, HIGH);
          digitalWrite(4, HIGH);
          delay(plen);
        };
        delay(plen);
        for (int x = 0; x < dyn; x++) {
          delay(plen);
          digitalWrite(myList[dyn - 1 - x] + 21, LOW);
          digitalWrite(4, LOW);
        };
      }; break;
  }
  if (remainingdur > 3 ) {
    delay((remainingdur - 3));
  };
}

// 6. Grace note cluster
// modedyn = 1 means use the given dyn ; modedyn = 2 means use a random dyn
// mode1 = 1 means end with silence; mode1 = 2 means end with note; mode1 = 3
means end with cluster
int gracenotesclust(int totd, int nbr, int dyn, int mode1, int modedyn) {
```

```
  int curdur;
  curdur = 55 * (1 + (random(100) / 100));
  for (int x = 1; x < nbr; x++) {
    if ( x % 2 < 1  ) {
      if ( modedyn < 2) {
        makenotedyn(curdur, dyn);
      } else {
        makenotedyn(curdur, random(1, 33));
      }
    }
    else {
      delay(curdur);
    }
  }
  switch (mode1) {
    case 1: {
        delay(totd - (nbr * curdur));
      }; break;
    case 2: {
        if ( modedyn < 2) {
          makenotedyn(totd - (nbr * curdur), dyn);
        } else {
          makenotedyn(totd - (nbr * curdur), random(1, 33));
        };
      }; break;
    case 3: {
        if ( modedyn < 2) {
          cluster(totd - (nbr * curdur), random(1, 3), random(1, 33));
        } else {
          makenotedyn(totd - (nbr * curdur), random(1, 33));
        };
      }; break;
  };
}

void gracenotesclust2(int totd, int nbr, int dyn, int mode1) {
  int curdur;
  curdur = 55 * (1 + (random(100) / 100));
  for (int x = 1; x < nbr; x++) {
    if ( x % 2 < 1  ) {
      makenotedyn(curdur, dyn);
    }
    else {
      delay(curdur);
    }
  }
  int curdurlast = (totd - (nbr * curdur));
  switch (mode1) {
```

```
    case 1: delay(curdurlast); break;
    case 2: makenotedyn(curdurlast, dyn); break;
    case 3: cluster(curdurlast, random(1, 3), dyn); break;
  };
}

void gracenotesclust2wt(int totd, int nbr, int dyn, int mode1) {
  int curdur;
  curdur = 55 * (1 + (random(100) / 100));
  for (int x = 1; x < nbr; x++) {
    if ( x % 2 < 1  ) {
      digitalWrite(4, HIGH); delay(1); digitalWrite(4, LOW);
      makenotedyn(curdur, dyn);
    }
    else {
      delay(curdur);
    }
  }
  int curdurlast = (totd - (nbr * curdur));
  switch (mode1) {
    case 1: delay(curdurlast); break;
    case 2: makenotedyn(curdurlast, dyn); break;
    case 3: cluster(curdurlast, random(1, 3), dyn); break;
  };
}

// 7. Constpulse
int constpulse(int totd, int dyn, int nbr, int variety) {
  int seeds[16];
  for (int x = 0; x < 16; x++) {
    seeds[x] = abs(random(999999));
  };
  for (int x = 1; x < (nbr + 1); x++) {
    for (int gr = 0; gr < variety; gr++) {
      makenotedyn(75, dyn);
      delay(((totd / nbr) - (75 * (1 + variety))));
    }
  }
}

void constpulse2(float totd, int dyn, int nbr, int notedur) {
  int restdur = round(((totd / nbr) - notedur));
  for (int x = 0; x < nbr; x++) {
    makenotedyn(notedur, dyn);
    delay(restdur);
  }
}
```

```
void constpulse2wt(float totd, int dyn, int nbr, int notedur) {
  int restdur = round(((totd / nbr) - notedur));
  for (int x = 0; x < nbr; x++) {
    digitalWrite(4, HIGH);
    makenotedyn(notedur, dyn);
    digitalWrite(4, LOW);
    delay(restdur);
  }
}

// 8. Noteattack
int noteattack(int totd, int nbr, int dyn, int fixed, int modedyn) {
  int curdur;
  curdur = 55 * (1 + (random(100) / 100));
  for (int x = 1; x < nbr; x++) {
    if ( x % 2 < 1  ) {
      if ( modedyn < 2) {
        makenotedyn(curdur, dyn);
      } else {
        makenotedyn(curdur, random(1, 33));
      }
    }
    else {
      delay(curdur);
    }
  }
  makenote2(totd - (nbr * curdur), fixed);
}

void noteattack2(int totd, int nbr, int dyn, int fixed) {
  int curdur;
  curdur = 55 * (1 + (random(100) / 100));
  for (int x = 1; x < nbr; x++) {
    if ( x % 2 < 1  ) {
      makenotedyn(curdur, dyn);
    }
    else {
      delay(curdur);
    }
  }
  makenote2(totd - (nbr * curdur), fixed);
}

void noteattack2wt(int totd, int nbr, int dyn, int fixed) {
  int curdur;
  curdur = 55 * (1 + (random(100) / 100));
  for (int x = 1; x < nbr; x++) {
    if ( x % 2 < 1  ) {
```

```
      digitalWrite(4, HIGH);
      makenotedyn(curdur, dyn);
      digitalWrite(4, LOW);
    }
    else {
      delay(curdur);
    }
  }
  digitalWrite(4, HIGH);
  makenote2(totd - (nbr * curdur), fixed);
  digitalWrite(4, LOW);
}

// 9. Acceloop
int acceloop(int variety, int nbrep, int mode, int initdur) {
  int nseq[16];  for (int x = 0; x < 16; x++) {
    nseq[x] = random(1, 33);
  };
  if ( mode < 2) {
    for (int k = 0; k < nbrep; k++) {
      int curdur = (initdur * pow(0.9, k));
      makenote2(75, nseq[k % variety]);
      delay(curdur);
    }
  }
  else {
    for (int k = nbrep; k > 0; k--) {
      int curdur = (1000 * pow(0.9, k));
      makenote2(75, nseq[k % variety]);
      delay(curdur);
    }
  }
}

int acceloopoly(int variety, int nbrep, int mode, int initdur) {
  int nseq[16];  for (int x = 0; x < 16; x++) {
    nseq[x] = random(1, 33);
  };
  int nseq2[16]; for (int x = 0; x < 16; x++) {
    nseq2[x] = random(1, 33);
  };
  int nseq3[16]; for (int x = 0; x < 16; x++) {
    nseq3[x] = random(1, 33);
  };
  int nseq4[16]; for (int x = 0; x < 16; x++) {
    nseq4[x] = random(1, 33);
  };
  if ( mode < 2) {
```

```
    for (int k = 0; k < nbrep; k++) {
      int curdur = (initdur * pow(0.9, k));
      makenote2(75, nseq[k % variety]);
      delay(curdur);
    }
  }
  else {
    for (int k = nbrep; k > 0; k--) {
      int curdur = (1000 * pow(0.9, k));
      int myChord[] = { nseq[k % variety], nseq2[(k + 1) % variety], nseq3[(k
+ 2) % variety], nseq4[(k + 3) % variety] };
      makechord(75, 4, myChord);
      delay(curdur);
    }
  }
}

int acceloopdurpoly(int variety, int nbrep, int mode, int globdur, int
notedur, int density) {

  int nseq[density][16];
  for (int y = 0; y < density; y++) {
    for (int x = 0; x < 16; x++) {
      nseq[y][x] = random(1, 33);
    };
  }

  int np2 = (globdur - (nbrep * notedur));
  float totp0 = 0.000;
  float pauses[nbrep];
  for (int k; k < nbrep; k++) {
    pauses[k] = (1.000 * pow(0.90, k));
    totp0 = (totp0 + pauses[k]);
  }

  int testdur = 0;
  for (int k = 0; k < nbrep; k++) {
    int curdur;
    if ( mode < 2) {
      curdur = round((pauses[k] * (np2 / totp0)));
    }
    else {
      curdur = round((pauses[nbrep - k - 1] * (np2 / totp0)));
    };

    int myChord[density];
    for (int y = 0; y < density; y++) {
      myChord[y] = nseq[y][((k + y) % variety)];
```

```
    }
    makechord(notedur, density, myChord);
    delay(curdur);
    testdur = (testdur + notedur + curdur);
  }
  if (testdur < globdur) {
    delay(globdur - testdur);
  };
}

int acceloopdurpolywt(int variety, int nbrep, int mode, int globdur, int
notedur, int density) {

  int nseq[density][16];
  for (int y = 0; y < density; y++) {
    for (int x = 0; x < 16; x++) {
      nseq[y][x] = random(1, 33);
    };
  }

  int np2 = (globdur - (nbrep * notedur));
  float totp0 = 0.000;
  float pauses[nbrep];
  for (int k; k < nbrep; k++) {
    pauses[k] = (1.000 * pow(0.90, k));
    totp0 = (totp0 + pauses[k]);
  }

  int testdur = 0;
  for (int k = 0; k < nbrep; k++) {
    int curdur;
    if ( mode < 2) {
      curdur = round((pauses[k] * (np2 / totp0)));
    }
    else {
      curdur = round((pauses[nbrep - k - 1] * (np2 / totp0)));
    };

    int myChord[density];
    for (int y = 0; y < density; y++) {
      myChord[y] = nseq[y][((k + y) % variety)];
    }
    digitalWrite(4, HIGH);
    makechord(notedur, density, myChord);
    digitalWrite(4, LOW);
    delay(curdur);
    testdur = (testdur + notedur + curdur);
  }
```

```
  if (testdur < globdur) {
    delay(globdur - testdur);
  };
}


// 10. Microstructure

int microstructure(int nbr) {
  switch (nbr) {
    case 1: wavnext(); cluster(random(500, 1000), random(1, 4), random(1, 6));
break;
    case 2: wavnext(); cluster(random(500, 5000), random(1, 4), random(1,
5));  break;
    case 3: wavnext(); constpulse(5000, random(1, 6), 10, random(1, 5));
break;
    case 4: wavnext(); constpulse(5000, random(1, 6), 10, 1);  break;
    case 5: wavnext(); acceloop(random(1, 12), random(30, 80), random(1, 3),
random(500, 1000)); break;
    case 6:
      wavnext();
      for (int k = 1; k < 2; k++) {
        int fixed1 = random(1, 33);
        for (int j = 1; j < 4; j++) {
          noteattack(5000, random(4, 64), 1, fixed1, random(1, 3));
          delay(100);
        }
      };  break;
    case 7:
      wavnext();
      int dns1[] = {1, 1, 2, 2, 3, 4, 5};
      wavnext();
      gracenotesclust(random(7000, 10000), random(5, 68), dns1[random(1, 7)],
random(1, 4), random(1, 3)); break;
  }
}


int microstructure2(int nbr) {
  switch (nbr) {
    case 1:  cluster(random(500, 1000), random(1, 4), random(1, 6)); break;
    case 2:  cluster(random(500, 5000), random(1, 4), random(1, 5));  break;
    case 3:  constpulse(5000, random(1, 6), 10, random(1, 5));  break;
    case 4:  constpulse(5000, (random(1, 6)), 10, 1);  break;
    case 5:  acceloop(random(1, 12), random(30, 80), random(1, 3), random(500,
1000)); break;
    case 6:
      for (int k = 1; k < 2; k++) {
```

```
        int fixed1 = random(1, 33);
        for (int j = 1; j < 4; j++) {
          noteattack(5000, random(4, 64), 1, fixed1, random(1, 3));
          delay(100);
        }
      };  break;
    case 7:
      int dns1[] = {1, 1, 2, 2, 3, 4, 5};
      gracenotesclust(random(7000, 10000), random(5, 68), dns1[random(1, 7)],
random(1, 4), random(1, 3)); break;
  }
}

int microstructure3(int nbr, int sndnbr) {
  switch (nbr) {
    case 1:  startsnd4(sndnbr, sndnbr, sndnbr, sndnbr); cluster(random(500,
1000), random(1, 4), random(1, 6)); break;
    case 2:  startsnd4(sndnbr, sndnbr, sndnbr, sndnbr); cluster(random(500,
5000), random(1, 4), random(1, 5));  break;
    case 3:  startsnd4(sndnbr, sndnbr, sndnbr, sndnbr); constpulse(5000,
random(1, 6), 10, random(1, 5));  break;
    case 4:  startsnd4(sndnbr, sndnbr, sndnbr, sndnbr); constpulse(5000,
(random(1, 6)), 10, 1);  break;
    case 5:  startsnd4(sndnbr, sndnbr, sndnbr, sndnbr); acceloop(random(1,
12), random(30, 80), random(1, 3), random(500, 1000)); break;
    case 6:
      startsnd4(sndnbr, sndnbr, sndnbr, sndnbr);
      for (int k = 1; k < 2; k++) {
        int fixed1 = random(1, 33);
        for (int j = 1; j < 4; j++) {
          noteattack(5000, random(4, 64), 1, fixed1, random(1, 3));
          delay(100);
        }
      };  break;
    case 7:
      startsnd4(sndnbr, sndnbr, sndnbr, sndnbr);
      int dns1[] = {1, 1, 2, 2, 3, 4, 5};
      gracenotesclust(random(7000, 10000), random(5, 68), dns1[random(1, 7)],
random(1, 4), random(1, 3)); break;
  }
  stopsnd4(sndnbr, sndnbr, sndnbr, sndnbr);
}

int microstructure4(int nbr) {
  switch (nbr) {
    case 1:  cluster(random(500, 1000), random(1, 4), random(1, 6)); break;
    case 2:  cluster(random(500, 5000), random(1, 4), random(1, 5));  break;
    case 3:  constpulse(5000, random(1, 6), 10, random(1, 5));  break;
```

```
    case 4:  constpulse(5000, (random(1, 6)), 10, 1);  break;
    case 5:  acceloop(random(1, 12), random(30, 80), random(1, 3), random(500,
1000)); break;
    case 6:
      for (int k = 1; k < 2; k++) {
        int fixed1 = random(1, 33);
        for (int j = 1; j < 4; j++) {
          noteattack(5000, random(4, 64), 1, fixed1, random(1, 3));
          delay(100);
        }
      };  break;
    case 7:
      int dns1[] = {1, 1, 2, 2, 3, 4, 5};
      gracenotesclust(random(7000, 10000), random(5, 68), dns1[random(0, 7)],
random(1, 4), random(1, 3)); break;
  }
}

int microstructure5(int nbr) {
  switch (nbr) {
    case 1:  cluster(random(500, 1000), random(1, 4), random(1, 8)); break;
    case 2:  cluster(random(500, 1000), random(1, 4), random(1, 5));  break;
    case 3:  constpulse(5000, random(1, 5), 10, random(1, 5));  break;
    case 4:  constpulse(10000, (random(1, 5)), 10, 1);  break;
    case 5:  acceloop(random(1, 16), random(20, 40), random(1, 3), random(400,
800)); break;
  }
}

// 11. Data Sections

int datasection1(int total, int bells[], int durations[]) {
  for (int k = 0; k < total; k++) {
    if (durations[k] < 0) {
      delay(abs(durations[k]));
    } else {
      makenote2(durations[k], bells[k]);
    }
  }
}

int datasection2(int total, int dur, int bells[], int durations[]) {
  for (int k = 0; k < total; k++) {
    makenote2(dur, bells[k]);
    delay(durations[k] - dur);
  }
}
```

```
int datasection3(int total, int bells[][32], int durations[]) {
  for (int k = 0; k < total; k++)   {
    for (int j = 0; j < 32; j++) {
      int el = bells[k][j];
      if (el > 0) {
        digitalWrite((el + 21), HIGH);
      }
    }
    delay(durations[k]);
    for (int j = 0; j < 32; j++) {
      int el = bells[k][j];
      if (el > 0) {
        digitalWrite((el + 21), LOW);
      }
    }
  }
}

int datasection4(int total, int dur, int bells[][32], int durations[]) {
  for (int k = 0; k < total; k++)   {
    for (int j = 0; j < 32; j++) {
      int el = bells[k][j];
      if (el > 0) {
        digitalWrite((el + 21), HIGH);
      }
    }
    delay(dur);
    for (int j = 0; j < 32; j++) {
      int el = bells[k][j];
      if (el > 0) {
        digitalWrite((el + 21), LOW);
      }
    }
    delay(durations[k] - dur);
  }
}

// 12. Note Structures (notestructure1 has noteon and noteoff commands)

int notestructure1(int total, int durs[], int bells[], int actions[]) {
  for (int k = 0; k < total; k++) {
    if (actions[k] > 0) {
      digitalWrite((bells[k] + 21), HIGH);
    }
    else {
      digitalWrite((bells[k] + 21), LOW);
    };
    delay(durs[k]);
```

```
  }
}

int notestrukt(int total, int strukt[][3]) {
  for (int k = 0; k < total; k++) {
    if (strukt[k][2] > 0) {
      digitalWrite((strukt[k][1] + 21), HIGH);
    }
    else {
      digitalWrite((strukt[k][1] + 21), LOW);
    };
    delay(strukt[k][0]);
  }
}

int notestructure2(int total, int durs[], int bells[][32], int actions[]) {
  for (int k = 0; k < total; k++) {
    if (actions[k] > 0) {
      for (int j = 0; j < 32; j++) {
        int el = bells[k][j];
        if (el > 0) {
          digitalWrite((el + 21), HIGH);
        }
      }
    }
    else {
      for (int j = 0; j < 32; j++) {
        int el = bells[k][j];
        if (el > 0) {
          digitalWrite((el + 21), LOW);
        }
      }
    };
    delay(durs[k]);
  }
}

// 12. Polyphonic Metric Fractions
int metric1(int total, int durations[], int unitdur, int polyphony) {

  int globdur = 0;
  for (int x = 0; x < total; x++) {
    globdur = (globdur + durations[x]);
  }

  int divall[polyphony][total];
  int r11all[polyphony][(32 * total)];
  int r11alllens[polyphony];
```

```
int nbrevents = 0;
for (int y = 0; y < polyphony; y++) {
  int curw = 0;
  int tots = 0;
  for (int x = 0; x < total; x++) {
    divall[y][x] = round(pow(2, (random(0, 500) / 100.0)));
    tots = (tots + divall[y][x]);
    for (int z = 0; z < (divall[y][x]); z++) {
      r11all[y][curw] = round(((float)durations[x] / (float)divall[y][x]));
      curw = (curw + 1);
    }
  };
  r11alllens[y] = tots;
  nbrevents = (nbrevents + (2 * r11alllens[y]));
};

Serial.println(nbrevents);

int dursall[polyphony][nbrevents];
int dursaccuall[nbrevents];
int actionsall[nbrevents];
int bellsall[nbrevents];

int curjj = 0;
int curjb = 0;
for (int y = 0; y < polyphony; y++) {
  int curtot = 0;
  for (int j = 0; j < (2 * r11alllens[y]) + 1; j++) {
    if ( j < (2 * r11alllens[y]) ) {
      if (j % 2 < 1) {
        dursall[y][j] = unitdur;
      }
      else {
        dursall[y][j] = ((r11all[y][j / 2]) - unitdur);
      };
      dursaccuall[curjj] = curtot;
      curtot = (curtot + dursall[y][j]);
      curjj = (curjj + 1);
    }
    else {
      dursaccuall[curjj] = curtot;
      curtot = (curtot + dursall[y][j]);
    };
  }
  for (int x = 0; x < total; x++) {
    int chn = random(1, 33);
    for (int j = 0; j < (2 * divall[y][x]); j++) {
```

```
        bellsall[curjb] = chn;
        if (j % 2 < 1) {
          actionsall[curjb] = 1;
        }
        else {
          actionsall[curjb] = 0;
        };
        curjb = (curjb + 1);
      }
    }
  }

  for (int i = 0; i < (nbrevents - 1); i++) {
    for (int o = 0; o < (nbrevents - (i + 1)); o++) {
      if (dursaccuall[o] > dursaccuall[o + 1]) {
        int t0 = dursaccuall[o];
        int t1 = bellsall[o];
        int t2 = actionsall[o];
        dursaccuall[o] = dursaccuall[o + 1];
        dursaccuall[o + 1] = t0;
        bellsall[o] = bellsall[o + 1];
        bellsall[o + 1] = t1;
        actionsall[o] = actionsall[o + 1];
        actionsall[o + 1] = t2;
      }
    }
  }

  for (int k = 0; k < nbrevents; k++) {
    if (actionsall[k] > 0) {
      digitalWrite((bellsall[k] + 21), HIGH);
    }
    else {
      digitalWrite((bellsall[k] + 21), LOW);
    };
    if (k < (nbrevents - 1)) {
      delay((dursaccuall[k + 1] - dursaccuall[k]));
    }
    else {
      delay((globdur - dursaccuall[k]));
    };
  }

}

void metric2(int duration, int unitdur, int polyphony) {

  int maxf1 = 500;
```

```
int divall[polyphony];
int r11all[polyphony][32];
int r11alllens[polyphony];

int nbrevents = 0;
for (int y = 0; y < polyphony; y++) {
  int curw = 0;
  int tots = 0;

  divall[y] = round(pow(2, (random(0, maxf1) / 100.0)));
  tots = (tots + divall[y]);
  for (int z = 0; z < (divall[y]); z++) {
    r11all[y][curw] = round(((float)duration / (float)divall[y]));
    curw = (curw + 1);
  }

  r11alllens[y] = tots;
  nbrevents = (nbrevents + (2 * r11alllens[y]));
};

int dursall[polyphony][nbrevents];
int dursaccuall[nbrevents];
int actionsall[nbrevents];
int bellsall[nbrevents];

int curjj = 0;
int curjb = 0;
for (int y = 0; y < polyphony; y++) {
  int curtot = 0;
  for (int j = 0; j < (2 * r11alllens[y]) + 1; j++) {
    if ( j < (2 * r11alllens[y]) ) {
      if (j % 2 < 1) {
        dursall[y][j] = unitdur;
      }
      else {
        dursall[y][j] = ((r11all[y][j / 2]) - unitdur);
      };
      dursaccuall[curjj] = curtot;
      curtot = (curtot + dursall[y][j]);
      curjj = (curjj + 1);
    }
    else {
      dursaccuall[curjj] = curtot;
      curtot = (curtot + dursall[y][j]);
    };
  }
```

```
    int chn = random(1, 33);
    for (int j = 0; j < (2 * divall[y]); j++) {
      bellsall[curjb] = chn;
      if (j % 2 < 1) {
        actionsall[curjb] = 1;
      }
      else {
        actionsall[curjb] = 0;
      };
      curjb = (curjb + 1);
    }

  }

  for (int i = 0; i < (nbrevents - 1); i++) {
    for (int o = 0; o < (nbrevents - (i + 1)); o++) {
      if (dursaccuall[o] > dursaccuall[o + 1]) {
        int t0 = dursaccuall[o];
        int t1 = bellsall[o];
        int t2 = actionsall[o];
        dursaccuall[o] = dursaccuall[o + 1];
        dursaccuall[o + 1] = t0;
        bellsall[o] = bellsall[o + 1];
        bellsall[o + 1] = t1;
        actionsall[o] = actionsall[o + 1];
        actionsall[o + 1] = t2;
      }
    }
  }

  for (int k = 0; k < nbrevents; k++) {
    if (actionsall[k] > 0) {
      digitalWrite((bellsall[k] + 21), HIGH);
    }
    else {
      digitalWrite((bellsall[k] + 21), LOW);
    };
    if (k < (nbrevents - 1)) {
      delay((dursaccuall[k + 1] - dursaccuall[k]));
    }
    else {
      delay((duration - dursaccuall[k]));
    };
  }

}

void metricharm(int duration, int unitdur, int polyphony, int bellstr[]) {
```

```
int maxf1 = 500;

int divall[polyphony];
int r11all[polyphony][32];
int r11alllens[polyphony];

int nbrevents = 0;
for (int y = 0; y < polyphony; y++) {
  int curw = 0;
  int tots = 0;

  divall[y] = (y + 1);
  tots = (tots + divall[y]);
  for (int z = 0; z < (divall[y]); z++) {
    r11all[y][curw] = round(((float)duration / (float)divall[y]));
    curw = (curw + 1);
  }

  r11alllens[y] = tots;
  nbrevents = (nbrevents + (2 * r11alllens[y]));
};

int dursall[polyphony][nbrevents];
int dursaccuall[nbrevents];
int actionsall[nbrevents];
int bellsall[nbrevents];

int curjj = 0;
int curjb = 0;
for (int y = 0; y < polyphony; y++) {
  int curtot = 0;
  for (int j = 0; j < (2 * r11alllens[y]) + 1; j++) {
    if ( j < (2 * r11alllens[y]) ) {
      if (j % 2 < 1) {
        dursall[y][j] = unitdur;
      }
      else {
        dursall[y][j] = ((r11all[y][j / 2]) - unitdur);
      };
      dursaccuall[curjj] = curtot;
      curtot = (curtot + dursall[y][j]);
      curjj = (curjj + 1);
    }
    else {
      dursaccuall[curjj] = curtot;
      curtot = (curtot + dursall[y][j]);
    };
```

```
    }

    int chn = bellstr[y];
    for (int j = 0; j < (2 * divall[y]); j++) {
      bellsall[curjb] = chn;
      if (j % 2 < 1) {
        actionsall[curjb] = 1;
      }
      else {
        actionsall[curjb] = 0;
      };
      curjb = (curjb + 1);
    }

  }

  for (int i = 0; i < (nbrevents - 1); i++) {
    for (int o = 0; o < (nbrevents - (i + 1)); o++) {
      if (dursaccuall[o] > dursaccuall[o + 1]) {
        int t0 = dursaccuall[o];
        int t1 = bellsall[o];
        int t2 = actionsall[o];
        dursaccuall[o] = dursaccuall[o + 1];
        dursaccuall[o + 1] = t0;
        bellsall[o] = bellsall[o + 1];
        bellsall[o + 1] = t1;
        actionsall[o] = actionsall[o + 1];
        actionsall[o + 1] = t2;
      }
    }
  }

  for (int k = 0; k < nbrevents; k++) {
    if (actionsall[k] > 0) {
      digitalWrite((bellsall[k] + 21), HIGH);
    }
    else {
      digitalWrite((bellsall[k] + 21), LOW);
    };
    if (k < (nbrevents - 1)) {
      delay((dursaccuall[k + 1] - dursaccuall[k]));
    }
    else {
      delay((duration - dursaccuall[k]));
    };
  }

}
```

```
void metric2wt(int duration, int unitdur, int polyphony) {

  int maxf1 = 500;

  int divall[polyphony];
  int r11all[polyphony][32];
  int r11alllens[polyphony];

  int nbrevents = 0;
  for (int y = 0; y < polyphony; y++) {
    int curw = 0;
    int tots = 0;

    divall[y] = round(pow(2, (random(0, maxf1) / 100.0)));
    tots = (tots + divall[y]);
    for (int z = 0; z < (divall[y]); z++) {
      r11all[y][curw] = round(((float)duration / (float)divall[y]));
      curw = (curw + 1);
    }

    r11alllens[y] = tots;
    nbrevents = (nbrevents + (2 * r11alllens[y]));
  };

  int dursall[polyphony][nbrevents];
  int dursaccuall[nbrevents];
  int actionsall[nbrevents];
  int bellsall[nbrevents];

  int curjj = 0;
  int curjb = 0;
  for (int y = 0; y < polyphony; y++) {
    int curtot = 0;
    for (int j = 0; j < (2 * r11alllens[y]) + 1; j++) {
      if ( j < (2 * r11alllens[y]) ) {
        if (j % 2 < 1) {
          dursall[y][j] = unitdur;
        }
        else {
          dursall[y][j] = ((r11all[y][j / 2]) - unitdur);
        };
        dursaccuall[curjj] = curtot;
        curtot = (curtot + dursall[y][j]);
        curjj = (curjj + 1);
      }
      else {
        dursaccuall[curjj] = curtot;
```

```
      curtot = (curtot + dursall[y][j]);
    };
  }

  int chn = random(1, 33);
  for (int j = 0; j < (2 * divall[y]); j++) {
    bellsall[curjb] = chn;
    if (j % 2 < 1) {
      actionsall[curjb] = 1;
    }
    else {
      actionsall[curjb] = 0;
    };
    curjb = (curjb + 1);
  }

}

for (int i = 0; i < (nbrevents - 1); i++) {
  for (int o = 0; o < (nbrevents - (i + 1)); o++) {
    if (dursaccuall[o] > dursaccuall[o + 1]) {
      int t0 = dursaccuall[o];
      int t1 = bellsall[o];
      int t2 = actionsall[o];
      dursaccuall[o] = dursaccuall[o + 1];
      dursaccuall[o + 1] = t0;
      bellsall[o] = bellsall[o + 1];
      bellsall[o + 1] = t1;
      actionsall[o] = actionsall[o + 1];
      actionsall[o + 1] = t2;
    }
  }
}

for (int k = 0; k < nbrevents; k++) {
  if (actionsall[k] > 0) {
    digitalWrite((bellsall[k] + 21), HIGH);
    digitalWrite(4, HIGH);
  }
  else {
    digitalWrite((bellsall[k] + 21), LOW);
    digitalWrite(4, LOW);
  };
  if (k < (nbrevents - 1)) {
    delay((dursaccuall[k + 1] - dursaccuall[k]));
    digitalWrite(4, LOW);
  }
  else {
```

```
      delay((duration - dursaccuall[k]));
      digitalWrite(4, LOW);
    };
  }

}

void metricaccel(float duration, int unitdur, int polyphony, int mode) {

  int maxf1;
  int maxf2 = round(10 * (log(duration / unitdur) / log(2)));
  if (maxf2 < 46) {
    maxf1 = maxf2;
  } else {
    maxf1 = 46;
  }
  int size11 = round(pow(2, (maxf1 / 10.0)));

  int divall[polyphony];
  int r11all[polyphony][size11];
  int r11alllens[polyphony];

  int nbrevents = 0;
  for (int y = 0; y < polyphony; y++) {
    int curw = 0;
    int tots = 0;
    int curz;
    divall[y] = round(pow(2, (random(0, maxf1) / 10.0)));

    float durfact = 0.0;
    for ( int uu = 0; uu < divall[y]; uu ++) {
      durfact = (durfact + ((1.0 / divall[y]) * (pow(0.9, uu))));
    }
    durfact = (1.0 / durfact);

    tots = (tots + divall[y]);
    for (int z = 0; z < (divall[y]); z++) {
      if (mode < 2) {
        curz = z;
      } else {
        curz = (divall[y] - 1 - z);
      };
      r11all[y][curw] = round(((duration * durfact) / (float)divall[y]) *
(pow(0.9, curz)));
      curw = (curw + 1);
    }

    r11alllens[y] = tots;
```

```
  nbrevents = (nbrevents + (2 * r11alllens[y]));
};

int dursall[polyphony][nbrevents];
int dursaccuall[nbrevents];
int actionsall[nbrevents];
int bellsall[nbrevents];

int curjj = 0;
int curjb = 0;
int testb;
for (int y = 0; y < polyphony; y++) {
  int curtot = 0;
  for (int j = 0; j < (2 * r11alllens[y]) + 1; j++) {
    if ( j < (2 * r11alllens[y]) ) {
      if (j % 2 < 1) {
        dursall[y][j] = unitdur;
      }
      else {
        testb = ((r11all[y][j / 2]) - unitdur);
        if (testb > 0) {
          dursall[y][j] = testb;
        }
        else
        {
          dursall[y][j] = 1;
        }
      };
      dursaccuall[curjj] = curtot;
      curtot = (curtot + dursall[y][j]);
      curjj = (curjj + 1);
    }
    else {
      dursaccuall[curjj] = curtot;
      curtot = (curtot + dursall[y][j]);
    };
  }

  int chn = random(1, 33);
  for (int j = 0; j < (2 * divall[y]); j++) {
    bellsall[curjb] = chn;
    if (j % 2 < 1) {
      actionsall[curjb] = 1;
    }
    else {
      actionsall[curjb] = 0;
    };
    curjb = (curjb + 1);
```

```
    }

  }

  for (int i = 0; i < (nbrevents - 1); i++) {
    for (int o = 0; o < (nbrevents - (i + 1)); o++) {
      if (dursaccuall[o] > dursaccuall[o + 1]) {
        int t0 = dursaccuall[o];
        int t1 = bellsall[o];
        int t2 = actionsall[o];
        dursaccuall[o] = dursaccuall[o + 1];
        dursaccuall[o + 1] = t0;
        bellsall[o] = bellsall[o + 1];
        bellsall[o + 1] = t1;
        actionsall[o] = actionsall[o + 1];
        actionsall[o + 1] = t2;
      }
    }
  }

  for (int k = 0; k < nbrevents; k++) {
    if (actionsall[k] > 0) {
      digitalWrite((bellsall[k] + 21), HIGH);
    }
    else {
      digitalWrite((bellsall[k] + 21), LOW);
    };
    if (k < (nbrevents - 1)) {
      delay((dursaccuall[k + 1] - dursaccuall[k]));
    }
    else {
      delay((duration - dursaccuall[k]));
    };
  }

}

// 13. random structure
void randstrukt(float totd, int divis, int dyn) {
  bubbleUnsort(myList, 32);
  int totstp1 = (totd / divis);
  int reststp = (totd - (totstp1 * divis));
  int onoff;

  for (int x = 0; x < totstp1; x++) {
    for (int y = 0; y < dyn; y++) {
      onoff = random(0, 2);
      if ( onoff < 1) {
```

```
        digitalWrite(myList[y] + 21, HIGH);
      }
      else {
        digitalWrite(myList[y] + 21, LOW);
      }
    }
    delay(divis);
  }
  delay(reststp);
  for (int y = 0; y < dyn; y++) {
    digitalWrite(myList[y] + 21, LOW);
  }
}

void randstruktwt(float totd, int divis, int dyn) {
  bubbleUnsort(myList, 32);
  int totstp1 = (totd / divis);
  int reststp = (totd - (totstp1 * divis));
  int onoff;

  for (int x = 0; x < totstp1; x++) {
    for (int y = 0; y < dyn; y++) {
      onoff = random(0, 2);
      if ( onoff < 1) {
        digitalWrite(myList[y] + 21, HIGH);
        digitalWrite(4, HIGH);
      }
      else {
        digitalWrite(myList[y] + 21, LOW);
        digitalWrite(4, LOW);
      }
    }
    delay(divis);
    digitalWrite(4, LOW);
  }
  delay(reststp);
  digitalWrite(4, LOW);
  for (int y = 0; y < dyn; y++) {
    digitalWrite(myList[y] + 21, LOW);
  }
}

// 14. Limits

void limits (int gld, int dur, int rate) {
  int bells[] = {26, 32, 34, 45, 47};
  int bellsize = 5;
  int ud = 0;
```

```
  for (int jj = 0; jj < (gld / dur); jj++) {
    ud = (4 + ((ud - 1 + random(0, 3)) % 4));
    for (int w = 0; w < (dur / rate); w++) {
      for (int  be = 0; be < bellsize; be++) {
        digitalWrite(bells[be], HIGH);
      };
      delay(ud);
      for (int  be = 0; be < bellsize; be++) {
        digitalWrite(bells[be], LOW);
      };
      delay(rate - ud);
    }
  }
}

void limits2(int gld, int dur, int rate) {
  int bells[] = {25, 28, 35, 40, 43, 46, 49, 50};
  int bellsize = 8;
  int ud = 0;
  for (int jj = 0; jj < (gld / dur); jj++) {
    ud = (4 + ((ud - 1 + random(0, 3)) % 4));
    for (int w = 0; w < (dur / rate); w++) {
      for (int  be = 0; be < bellsize; be++) {
        digitalWrite(bells[be], HIGH);
      };
      delay(ud);
      for (int  be = 0; be < bellsize; be++) {
        digitalWrite(bells[be], LOW);
      };
      delay(rate - ud);
    }
  }
  int finaldelay1 = (gld / dur);
  int finaldelay2 = (dur / rate);
  int finaldelay3 = (gld - (finaldelay1 * finaldelay2 * rate));
  if (finaldelay3 > 0) {
     Serial.print("delay ="); Serial.println(finaldelay3);
     delay(finaldelay3);
  };
}

void limits3(int gld, int dur, int rate, int bellslimit[], int belllimitsize) {
  int ud = 0;
  for (int jj = 0; jj < (gld / dur); jj++) {
    ud = (4 + ((ud - 1 + random(0, 3)) % 4));
    for (int w = 0; w < (dur / rate); w++) {
      for (int  be = 0; be < belllimitsize; be++) {
        digitalWrite(21 + bellslimit[be], HIGH);
```

```
      };
      delay(ud);
      for (int  be = 0; be < belllimitsize; be++) {
        digitalWrite(21 + bellslimit[be], LOW);
      };
      delay(rate - ud);
    }
  }
  int finaldelay1 = (gld / dur);
  int finaldelay2 = (dur / rate);
  int finaldelay3 = (gld - (finaldelay1 * finaldelay2 * rate));
  // if(finaldelay3 > 0) { delay(finaldelay3); };
}

void limits4(int gld, int dur, int rate, int bellslimit[], int belllimitsize) {
  int ud = 0;
  for (int jj = 0; jj < (gld / dur); jj++) {
    ud = (4 + ((ud - 1 + random(0, 3)) % 4));
    for (int w = 0; w < (dur / rate); w++) {
      for (int  be = 0; be < belllimitsize; be++) {
        digitalWrite(21 + bellslimit[be], HIGH);
      };
      delay(ud);
      for (int  be = 0; be < belllimitsize; be++) {
        digitalWrite(21 + bellslimit[be], LOW);
      };
      delay(rate - ud);
    }
  }
  int finaldelay1 = (gld / dur);
  int finaldelay2 = (dur / rate);
  int finaldelay3 = (gld - (finaldelay1 * finaldelay2 * rate));
  if (finaldelay3 > 0) {
    delay(finaldelay3);
  };
}

// DEFINE A MICROSTRUCTURE THAT HAS DURATION AS AN INPUT PARAMETER

void microstructuredur(int type, int duration, int dyn) {
  switch (type) {
    case 1: cluster2(duration, random(1, 4), dyn); break;
    case 2: acceloopdurpoly(random(1, 12), random(20, 80), random(1, 3),
duration, random(40, 75), dyn); break;
    case 3: metric2(duration, random(50, 75), round(pow(2, (random(0, 350) /
100.0)))); break;
    case 4: constpulse2(duration, dyn, random(2, 20), random(50, 75)); break;
    case 5: noteattack2(duration, round(pow(2, (random(200, 601) / 100.0))),
```

```
dyn, random(1, 33)); break;
    case 6: gracenotesclust2(duration, random(5, 68), dyn, random(1, 4));
break;
    case 7: notedyn(duration, dyn, round(pow(2.154, (random(100, 601) / 100.
0)))); break;
    case 8: randstrukt(duration, random(5, (duration / 3)), dyn); break;
    case 9: metricaccel(duration, random(35, 45), random(1, 13), random(1,
3)); break;
    case 10: limits2(duration, 200, random(15, 40)); break;
    case 11: delay(duration);
  }
}

// DEFINE A MICROSTRUCTURE THAT HAS DURATION AS AN INPUT PARAMETER + WAVTRIGGER

void microstructuredurwt(int type, int duration, int dyn) {
  switch (type) {
    case 1: cluster2wt(duration, random(1, 4), dyn);
      break;
    case 2: acceloopdurpolywt(random(1, 12), random(20, 80), random(1, 3),
duration, random(40, 75), dyn);
      break;
    case 3: metric2wt(duration, random(50, 75), round(pow(2, (random(0, 350) /
100.0))));
      break;
    case 4: constpulse2wt(duration, dyn, random(2, 20), random(50, 75));
      break;
    case 5: noteattack2wt(duration, round(pow(2, (random(200, 601) / 100.0))),
dyn, random(1, 33));
      break;
    case 6: gracenotesclust2wt(duration, random(5, 68), dyn, random(1, 4));
      break;
    case 7: wavnext(); notedyn(duration, dyn, round(pow(2.154, (random(100,
601) / 100.0))));
      break;
    case 8: randstruktwt(duration, random(5, (duration / 5)), dyn);
      break;
    case 9: delay(duration);
  }
}


// ---------------------------------------------------------------------- //
// ----------------------------VOID SETUP---------------------------- //

void setup () {

  delay(800);
```

```
  randomSeed(1);

  Serial.begin(9600);
  mp3_set_serial(Serial);      //set Serial for DFPlayer-mini mp3 module
  delay(1);                    // delay 1ms to set volume
  mp3_set_volume(30);         // value 0~30
  Serial1.begin(9600);
  Serial2.begin(9600);
  Serial3.begin(9600);

  for ( int sndnbr = 1; sndnbr < 501; sndnbr++) {
    stopsnd4(sndnbr, sndnbr, sndnbr, sndnbr);
  };
  wavstop();

  for (int x = 2; x < 54; x++) {
    pinMode(x, OUTPUT);
  };
  for (int x = 2; x < 54; x++) {
    digitalWrite(x, LOW);
  };
  delay(137);
}

unsigned long time1;
int realminutes;

// ------------------------------------------------------------------------ //
// ----------------------------VOID LOOP---------------------------- //

void loop() {
  for ( int sndnbr = 1; sndnbr < 501; sndnbr++) {
    stopsnd4(sndnbr, sndnbr, sndnbr, sndnbr);
  };
  randomSeed(1);
  Serial.println(); Serial.println();
  Serial.println("-----START------");
  Serial.println();

  //
-------------------------------------------------------------------------------
-
  // PARTE 1
  Serial.println(); Serial.println("- PART I - intro");
  Serial.print("Time: "); time1 = millis(); Serial.println(time1 - 4000);
  Serial.print("Time: "); realminutes = (time1 / 60000); Serial.
print(realminutes); Serial.print(":"); Serial.print(((time1 - (realminutes *
60000)) / 1000));
```

```
  wavnormal();

  delay(9600);

  wavnext();
  gracenotesclust(random(8000, 10000), random(10, 68), 4, random(1, 4), 1);
  acceloopoly(16, random(20, 40), 2, 300);

  wavnext();
  acceloopoly(3, random(30, 60), 1, 400);

  for (int x = 1; x < 2; x++) {
    int dns[] = {1, 1, 2, 2, 3, 4, 5};
    wavnext();
    gracenotesclust(random(8000, 10000), random(5, 68), dns[random(1, 7)],
random(1, 4), 1);
  }
  wavnext(); acceloop(16, random(30, 80), 2, random(400, 800));

  // 1b. GRACE-NOTES  com Wav trigger
  Serial.println(); Serial.println("- PART Ib -");
  Serial.print("Time: "); time1 = millis(); Serial.println(time1 - 4000);
  Serial.print("Time: "); realminutes = (time1 / 60000); Serial.
print(realminutes); Serial.print(":"); Serial.println(((time1 - (realminutes *
60000)) / 1000));

  wavnext(); for (int k = 1; k < 2; k++) {
    int fixed1 = 15;
    for (int j = 1; j < 4; j++) {
      noteattack(3333, random(8, 34), 2, fixed1, random(1, 3));
      delay(100);
    }
  }
  wavnext();
  delay(1000);
  // constpulse(10000, 3, 10, 11);
  for (int x = 1; x < 3; x++) {
    int dns[] = {1, 1, 2, 2, 3, 4, 5}; if (x > 1) {
      wavnext();
    };
    gracenotesclust(random(8000, 9000), random(3, 16), dns[random(4, 7)],
random(1, 2), random(1, 3));
  }

  // 8 wavnext's
  //
-------------------------------------------------------------------------
```

```
-
  // PARTE 2
  // 2.1. CLUSTERS
  Serial.println(); Serial.println("- PART II - single shot clusters");
  Serial.print("Time: "); time1 = millis(); Serial.println(time1 - 4000);
  Serial.print("Time: "); realminutes = (time1 / 60000); Serial.
print(realminutes); Serial.print(":"); Serial.println(((time1 - (realminutes *
60000)) / 1000));

  for (int x = 1; x < 5; x++) {
    if (x < 2) {
      wavstop();
    }
    wavnext();
    cluster(550 * random(1, 3), 1, random(10, 33));
    delay(725 * (random(40, 100) / 10));
  }

  delay(3600);

  // 13 wavnext's
  //
-------------------------------------------------------------------------
-
  // PARTE 3
  Serial.println(); Serial.println("- PART III - continuation");
  Serial.print("Time: "); time1 = millis(); Serial.println(time1 - 4000);
  Serial.print("Time: "); realminutes = (time1 / 60000); Serial.
print(realminutes); Serial.print(":"); Serial.println(((time1 - (realminutes *
60000)) / 1000));

  wavnext();
  constpulse(10000, 2, 10, 1);
  for (int x = 1; x < 10; x++) {

    microstructure(random(1, 8));
    delay(random(5, 500));

    if ( x > 8) {
      onsetsnd4(102, 104, 105, 106, 50, 1000, 1000, 1000);
    };

  }
  delay(10000);
  metric2(10000, 18, 3);
  delay(5000);
  stopsnd4(102, 104, 105, 106);
```

```
  //
-------------------------------------------------------------------------
-
  // PARTE 4
  Serial.println(); Serial.println("- PART IV - single shots");
  Serial.print("Time: "); time1 = millis(); Serial.println(time1 - 4000);
  Serial.print("Time: "); realminutes = (time1 / 60000); Serial.
print(realminutes); Serial.print(":"); Serial.println(((time1 - (realminutes *
60000)) / 1000));
  stopsnd4(102, 104, 105, 106);
  // onsetsnd4(9, 65, 29, 50, 1, 1, 1, 1);
  wavstop();
  wavnext();
  cluster(500 * random(1, 3), 1, random(1, 33));
  delay(6400);

  delay(100);
  // stopsnd4(9,65,29,50);
  onsetsnd4(303,303, 303, 303,10,5,10,5);
  delay(7000);
  //metric2(11000, 22, 7);

  //
-------------------------------------------------------------------------
-
  // PARTE 4b - moving masses
  Serial.println(); Serial.println("- PART IVb - moving masses");
  Serial.print("Time: "); time1 = millis(); Serial.println(time1 - 4000);
  Serial.print("Time: "); realminutes = (time1 / 60000); Serial.
print(realminutes); Serial.print(":"); Serial.println(((time1 - (realminutes *
60000)) / 1000));

  int bel11[22];
  int neworder2[32];
  for (int www = 0; www < 32; www++) {
    neworder2[www] = www + 1;
  };
  bubbleUnsort(neworder2, sizeof(neworder2) / sizeof(int));
  for (int w = 0; w < 16; w++) {
    Serial.println(w);
    for (int ww = (w); ww < 8 + (w); ww++) {
      bel11[ww - (w)] = 21 + neworder2[ww];
    };
    limits3((20 + (2*w))*52, (20 + (2*w))*4, 20 + (2*w), bel11, (2*w)+1);
    delay(pow(2,random(2,11)));
  }

  delay(9000);
```

```
// ----------------
onsetsnd4(9, 65, 50, 29, 30, 50, 80, 130);
wavnext();
cluster(1100, 1, random(26, 33));
delay(725 * (random(40, 100) / 10));
delay(6000);
// ----------------
onsetsnd4(500, 500, 500, 500, 30, 50, 80, 130);
wavnext();
delay(500);
cluster(320 * random(1, 3), 3, random(20, 33));
metric2(6500, 75, 8);
metricaccel(5200, 50, 6, 1);

int bel13[23];
int neworder40[32];
for (int www = 0; www < 32; www++) {
  neworder40[www] = www + 1;
};
bubbleUnsort(neworder40, sizeof(neworder40) / sizeof(int));
bubbleUnsort(neworder40, sizeof(neworder40) / sizeof(int));
bubbleUnsort(neworder40, sizeof(neworder40) / sizeof(int));
for (int wer = 0; wer < 5; wer++) {
  for (int ww = (wer); ww < 23 + (wer); ww++) {
    bel13[ww - (wer)] = neworder40[ww];
  };
  if (wer < 4) {
    limits4(5000, 200, random(20, 50), bel13, 23);
  }
  else {
    limits4(3300, 200, random(40, 70), bel13, 23);
  }
}
delay(250);
makenotedyn(random(50) + 50, random(15, 33));
delay(50);
makenotedyn(random(100) + 50, random(15, 33));
delay(100);
makenotedyn(random(150) + 50, random(15, 33));
delay(50);
makenotedyn(random(200) + 50, random(15, 33));
metric2(12000, 70, 8);
// delay(3000);
cluster(320 * random(1, 3), 1, random(20, 33));

for (int x = 1; x < 3; x++) {

  if ( x < 2)
```

```
    { // wavstop();
      wavnext();
      cluster(500 * random(1, 3), 1, random(15, 33));
      delay(900 * (random(40, 100) / 10));
    }
    else {
      // wavstop();
      wavnext();
      delay(500);
      cluster(500 * random(1, 3), 1, random(15, 33));
      delay(900 * (random(40, 100) / 10));
      cluster(500 * random(1, 3), 3, random(15, 33));
    } ;

  }
  limits2(7000, 200, 40);

  wavnext();
  delay(1100);
  cluster(500 * random(1, 3), 1, random(15, 33));
  delay(500 * (random(40, 100) / 10));

  for (int x = 1; x < 3; x++) {
    wavnext();
    if ( x < 2) {
      delay(1000);
    } else {
      delay(150);
    }
    cluster(500 * random(1, 3), 1, random(15, 33));
    delay(1000);
  }
  limits2(8000, 600, 20);
  limits2(3000, 600, 20);
  //
----------------------------------------------------------------------------
-
  // PARTE 4c - regular rhythms harmonic strucuture
  Serial.println(); Serial.println("- PART VII - regular rhythms harmonic
strucuture");
  Serial.print("Time: "); time1 = millis(); Serial.println(time1 - 4000);
  Serial.print("Time: "); realminutes = (time1 / 60000); Serial.
print(realminutes); Serial.print(":"); Serial.println(((time1 - (realminutes *
60000)) / 1000));
  wavstop();

  int smd = 7;
  int startx1=random(1,31);
```

```
  for (int x = 1; x < 33; x++) { makenotedyn(smd, x); delay(100-smd); };
  for (int x = 32; x > 0; x--) { makenotedyn(smd, x); delay(100-smd); };
  for (int x = 1; x < 33; x++) { makenotedyn(smd, x); delay(100-smd); };
  for(int x=0; x < 160; x++) {
    makenotedyn(smd, startx1);
    delay(100-smd);
    startx1 = (2+((startx1+random(0,3)-1)%31));
   };


  int neworder[32];
  for (int www = 0; www < 32; www++) {
    neworder[www] = www + 1;
  };
  bubbleUnsort(neworder, sizeof(neworder) / sizeof(int));

  int durd1 = 1000;
  int maxi1 = 8;
  int smald1 = 40;

  for ( int ww = 0; ww < maxi1; ww++) {
    Serial.println(ww + 1);
    for ( int w = 0; w < random(3, 5); w++) {
      metricharm(durd1, smald1, ww + 1, neworder);
    }
  }

  //
-------------------------------------------------------------------------
-
  // PARTE 5
  Serial.println(); Serial.println("- PART V - Gruta");
  Serial.print("Time: "); time1 = millis(); Serial.println(time1 - 4000);
  Serial.print("Time: "); realminutes = (time1 / 60000); Serial.
print(realminutes); Serial.print(":"); Serial.println(((time1 - (realminutes *
60000)) / 1000));
  // wavstop();

  wavnext();
  makenotedyn(random(50) + 50, random(10, 33));
  for (int x = 1; x < 4; x++) {
    int sndnbr1 = random(100, 104);
    int sndnbr2 = random(100, 104);
    int sndnbr3 = random(100, 104);
    int sndnbr4 = random(100, 104);
    startsnd4(sndnbr1, sndnbr2, sndnbr3, sndnbr4);
    limits(7000, 200, 30);
```

```
    int bel12[16];
    int neworder3[32];
    for (int www = 0; www < 32; www++) {
      neworder3[www] = www + 1;
    };
    bubbleUnsort(neworder3, sizeof(neworder3) / sizeof(int));
    limits3(random(3500, 7000), 200, random(20, 70), neworder3, 16);
    if ( x < 3) {
      metric2(random(4000, 8000), random(15, 55), random(4,8));
    } else {
      limits2(random(3000, 6000), 200, random(15, 40));
    };
    delay(random(0,500));
    metric2(random(3000, 6000), random(40, 75), 6);
    delay(random(0,500));
    limits2(random(3000, 6000), 200, random(20, 50));
    stopsnd4(sndnbr1, sndnbr2, sndnbr3, sndnbr4);
  }

  // wavstop();
  //
-------------------------------------------------------------------------
-
  // PARTE 6
  Serial.println(); Serial.println("- PART VI - Auscultadores");
  Serial.print("Time: "); time1 = millis(); Serial.println(time1 - 4000);
  Serial.print("Time: "); realminutes = (time1 / 60000); Serial.
print(realminutes); Serial.print(":"); Serial.println(((time1 - (realminutes *
60000)) / 1000));
  wavnext();

  limits2(random(4000, 6500), 200, random(20, 60));

  for (int x = 1; x < 28; x++) {
    mp3_set_serial (Serial); mp3_play (random(200, 275)); delay(1);
    mp3_set_serial (Serial1); mp3_play (random(200, 275)); delay(1);
    mp3_set_serial (Serial2); mp3_play (random(200, 275)); delay(1);
    mp3_set_serial (Serial3); mp3_play (random(200, 275)); delay(1);
    if (x % 8 < 1) {
      microstructuredur(random(1, 9), random(4000, 5000), round(pow(2,
(random(0, 501) / 100.0))));
    }
    else {
      makenotedyn(random(50) + 50, random(15, 33));
    };
    int riri1 = round(pow(2, (random(0, 1200) / 100.0)));
    mp3_set_serial (Serial); mp3_play (random(200, 275)); delay(1);
    mp3_set_serial (Serial1); mp3_play (random(200, 275)); delay(1);
```

```
    mp3_set_serial (Serial2); mp3_play (random(200, 275)); delay(1);
    mp3_set_serial (Serial3); mp3_play (random(200, 275)); delay(1);
    delay(riri1);
  }
  limits2(random(7000, 14000), 200, random(20, 60));
  // acceloopdurpoly(random(1, 12), random(25, 80), 2, 20000, random(50, 75),
round(pow(2, (random(0, 151) / 100.0))));


  //
----------------------------------------------------------------------------
-
  // PARTE 7

  int smd2 = 6;
  int startx2=random(1,31);
  for (int x = 1; x < 33; x++) { makenotedyn(smd2, x); delay(100-smd2); };
  for (int x = 32; x > 0; x--) { makenotedyn(smd2, x); delay(100-smd2); };
  for (int x = 1; x < 33; x++) { makenotedyn(smd2, x); delay(100-smd2); };
  for(int x=0; x < 300; x++) {
    makenotedyn(smd2, startx2);
    delay(100-smd2);
    startx2 = (2+((startx2+random(0,3)-1)%31));
   };

  for ( int sndnbr = 200; sndnbr < 276; sndnbr++) {
    stopsnd4(sndnbr, sndnbr, sndnbr, sndnbr);
  };

  //
----------------------------------------------------------------------------
-
  // PARTE 8

  randomSeed(111111111111);
  // walkforward(32); wavstop(); delay(3000); wavstop();
  Serial.println(); Serial.println("- PART VIII - new experiments");
  Serial.print("Time: "); time1 = millis(); Serial.println(time1 - 4000);
  Serial.print("Time: "); realminutes = (time1 / 60000); Serial.
print(realminutes); Serial.print(":"); Serial.println(((time1 - (realminutes *
60000)) / 1000));
  wavstop();
  wavnext();

  int posstyp[] = {1, 2, 3, 3, 3, 4, 5, 6, 7, 8, 9, 10};
  int posstypsmall[] = {1, 7, 8, 9};
  int realtp;
  int du1[28] = {14900, 5500, 5700, 10900, 900, 800, 6000, 10000, 9700,
```

```
8000,8000, 5800, 14300,
                 15000,4050, 14300, 14750, 8700, 17700, 9750, 8300, 11450,
8000 ,8500, 12000,12250, 7120, 11160
                };
  unsigned long time2a;
  unsigned long time2b;
  int mtype;

  startsnd4(9,65,29,50); metric2(du1[0], 40, 3);
  mtype = 3; Serial.println(1); Serial.print("mtype = "); Serial.
println(mtype); time2a = millis(); startanysnd4(); microstructuredur(mtype,
du1[1], 5); time2b = millis(); Serial.println(time2b - time2a);
  mtype = posstyp[random(0, 11)]; Serial.println(2); Serial.print("mtype = ");
Serial.println(mtype); time2a = millis(); startanysnd4();
microstructuredur(mtype, du1[2], 5); time2b = millis(); Serial.println(time2b
- time2a);
  mtype = posstyp[random(0, 11)]; Serial.println(3); Serial.print("mtype = ");
Serial.println(mtype); time2a = millis(); startanysnd4();
microstructuredur(mtype, du1[3], 16); time2b = millis(); Serial.println(time2b
- time2a);
  mtype = 1; Serial.println(4); Serial.print("mtype = "); Serial.
println(mtype); time2a = millis(); startanysnd4(); microstructuredur(mtype,
du1[4], 12); time2b = millis(); Serial.println(time2b - time2a);
  mtype = 1; Serial.println(5); Serial.print("mtype = "); Serial.
println(mtype); time2a = millis(); startanysnd4(); microstructuredur(mtype,
du1[5], 24); time2b = millis(); Serial.println(time2b - time2a);
  mtype = posstyp[random(0, 11)]; Serial.println(6); Serial.print("mtype = ");
Serial.println(mtype); time2a = millis(); startanysnd4();
microstructuredur(mtype, du1[6], round(pow(2, (random(0, 501) / 100.0))));
time2b = millis(); Serial.println(time2b - time2a);
  mtype = posstyp[random(0, 11)]; Serial.println(7); Serial.print("mtype = ");
Serial.println(mtype); time2a = millis(); startanysnd4();
microstructuredur(mtype, du1[7], round(pow(2, (random(200, 501) / 100.0))));
time2b = millis(); Serial.println(time2b - time2a);
  wavnext(); mtype = posstyp[random(0, 11)]; Serial.println(8); Serial.
print("mtype = "); Serial.println(mtype); time2a = millis(); startanysnd4();
microstructuredur(mtype, du1[8], round(pow(2, (random(0, 501) / 100.0))));
time2b = millis(); Serial.println(time2b - time2a);
  startanysnd4(); limits2( du1[9], 200, random(15, 40));
  wavnext(); mtype = posstyp[random(0, 11)]; Serial.println(10); Serial.
print("mtype = "); Serial.println(mtype); time2a = millis(); startanysnd4();
microstructuredur(mtype, du1[10], round(pow(2, (random(0, 501) / 100.0))));
time2b = millis(); Serial.println(time2b - time2a);
  mtype = posstyp[random(0, 11)]; Serial.println(11); Serial.print("mtype =
"); Serial.println(mtype); time2a = millis(); startanysnd4();
microstructuredur(mtype, du1[11], round(pow(2, (random(0, 501) / 100.0))));
time2b = millis(); Serial.println(time2b - time2a);
  startanysnd4(); limits2( du1[12], 200, random(15, 40));
```

```
  wavnext(); startsnd4(9,65,29,50); mtype = 3; Serial.println(13); Serial.
print("mtype = "); Serial.println(mtype); time2a = millis();
microstructuredur(mtype, du1[13], round(pow(2, (random(0, 501) / 100.0))));
time2b = millis(); Serial.println(time2b - time2a);
  mtype = 8; Serial.println(14); Serial.print("mtype = "); Serial.
println(mtype); time2a = millis(); startanysnd4(); microstructuredur(mtype,
du1[14], round(pow(2, (random(0, 501) / 100.0)))); time2b = millis(); Serial.
println(time2b - time2a);
  wavnext(); mtype = posstyp[random(0, 11)]; Serial.println(15); Serial.
print("mtype = "); Serial.println(mtype); time2a = millis(); startanysnd4();
microstructuredur(mtype, du1[15], round(pow(2, (random(0, 501) / 100.0))));
time2b = millis(); Serial.println(time2b - time2a);
  startanysnd4(); limits2( du1[16], 200, random(15, 40));
  wavnext(); startsnd4(9,65,29,9); mtype = posstyp[random(0, 11)]; Serial.
println(17); Serial.print("mtype = "); Serial.println(mtype); time2a =
millis(); microstructuredur(mtype, du1[17], round(pow(2, (random(0, 501) / 100.
0)))); time2b = millis(); Serial.println(time2b - time2a);
  wavnext(); mtype = posstyp[random(0, 11)]; Serial.println(18); Serial.
print("mtype = "); Serial.println(mtype); time2a = millis(); startanysnd4();
microstructuredur(mtype, du1[18], round(pow(2, (random(0, 501) / 100.0))));
time2b = millis(); Serial.println(time2b - time2a);
  startanysnd4(); limits2( du1[19], 200, random(15, 40));
  wavnext(); mtype = posstyp[random(0, 11)]; Serial.println(20); Serial.
print("mtype = "); Serial.println(mtype); time2a = millis(); startanysnd4();
microstructuredur(mtype, du1[20], round(pow(2, (random(0, 501) / 100.0))));
time2b = millis(); Serial.println(time2b - time2a);
  wavnext(); mtype = posstyp[random(0, 11)]; Serial.println(21); Serial.
print("mtype = "); Serial.println(mtype); time2a = millis(); startanysnd4();
microstructuredur(mtype, du1[21], round(pow(2, (random(0, 501) / 100.0))));
time2b = millis(); Serial.println(time2b - time2a);
  wavnext(); mtype = 3; Serial.println(22); Serial.print("mtype = "); Serial.
println(mtype); time2a = millis(); startanysnd4(); microstructuredur(mtype,
du1[22], round(pow(2, (random(0, 501) / 100.0)))); time2b = millis(); Serial.
println(time2b - time2a);
  mtype = 3; Serial.println(23); Serial.print("mtype = "); Serial.
println(mtype); time2a = millis(); startanysnd4(); microstructuredur(mtype,
du1[23], round(pow(2, (random(0, 501) / 100.0)))); time2b = millis(); Serial.
println(time2b - time2a);
  wavnext(); mtype = posstyp[random(0, 11)]; Serial.println(24); Serial.
print("mtype = "); Serial.println(mtype); time2a = millis(); startanysnd4();
microstructuredur(mtype, du1[24], round(pow(2, (random(0, 501) / 100.0))));
time2b = millis(); Serial.println(time2b - time2a);
  mtype = posstyp[random(0, 11)]; Serial.println(25); Serial.print("mtype =
"); Serial.println(mtype); time2a = millis(); startanysnd4();
microstructuredur(mtype, du1[25], round(pow(2, (random(0, 501) / 100.0))));
time2b = millis(); Serial.println(time2b - time2a);
  wavnext(); startanysnd4(); limits2( du1[26], 200, random(15, 40));
  wavnext(); acceloopdurpoly(random(1, 12), random(25, 80), 2, (du1[27]+1000),
```

```
random(50, 75), round(pow(2, (random(0, 151) / 100.0))));
  // mtype = posstyp[random(0, 11)]; Serial.println(27); Serial.print("mtype =
"); Serial.println(mtype); time2a = millis();  microstructuredur(mtype,
du1[27], round(pow(2, (random(0, 501) / 100.0)))); time2b = millis(); Serial.
println(time2b - time2a);

  delay(7500);
  wavstop();
  delay(1000);

  /*
  //
----------------------------------------------------------------------------
-
  // PARTE 9 - regular rhythms harmonic strucuture
  Serial.println(); Serial.println("- PART IX - regular rhythms harmonic
strucuture");
  Serial.print("Time: "); time1 = millis(); Serial.println(time1 - 4000);
  Serial.print("Time: "); realminutes = (time1 / 60000); Serial.
print(realminutes); Serial.print(":"); Serial.println(((time1 - (realminutes *
60000)) / 1000));

  int neworder[32];
  for (int www = 0; www < 32; www++) {
    neworder[www] = www + 1;
  };
  bubbleUnsort(neworder, sizeof(neworder) / sizeof(int));

  int durd1 = 2000;
  int maxi1 = 9;
  int smald1 = 40;

  for ( int ww = 0; ww < maxi1; ww++) {
    Serial.println(ww + 1);
    for ( int w = 0; w < random(2, 5); w++) {
      metricharm(durd1, smald1, ww + 1, neworder);
    }
  }

  for ( int ww = maxi1; ww > 0; ww--) {
    Serial.println(ww + 1);
    for ( int w = 0; w < random(2, 5); w++) {
      metricharm(durd1, smald1, ww, neworder);
    }
  }

  for (int kk = 0; kk < 16; kk++) {
    int div1 = round(pow(2, (random(0, 350) / 100.0)));
```

```
    Serial.println(div1);
    for (int jjj = 0; jjj < random(1, 5); jjj++) {
      metricharm(durd1, smald1, div1, neworder);
    }
  }

  //
-------------------------------------------------------------------------------
-
  // PARTE 10 - test microstructuredurwt
  /*
    walkforward(33); wavstop(); delay(6000); wavstop();

    microstructuredurwt(4, 8000, 8); wavstop();
    microstructuredurwt(4, 8000, 4); wavstop();
    microstructuredurwt(4, 8000, 2); wavstop();
    microstructuredurwt(4, 8000, 3); wavstop();
    microstructuredurwt(4, 8000, 6); wavstop();

    for(int k=0; k < 40; k++) {
    wavnext();
    notedyn(4000, 3, 5);
    wavstop();
    }

    for (int j=1; j< 10; j++) { microstructuredurwt(j, 16000, 4); wavstop(); }



    microstructuredurwt(3, 8000, 6);
    microstructuredurwt(3, 8000, 8);
    microstructuredurwt(8, 8000, 8);
    microstructuredurwt(8, 8000, 8);

    wavstop();
    delay(20000);
  */

  Serial.println("END");
  Serial.println();

}
```