Annex 1: Programa

1. Codi

```
#include <Freq.h>
#include <Tone.h>
#include <avr/pgmspace.h>
#include <PROGMEM_readAnything.h>
#include <notesThereminPROGMEM.h>
#include <MelodiesTocataPROGMEM.h>
#include <LowPower.h>


//Polsadors i LED polsadors
bool play = false;    //false -> Sleep ; true -> Play
bool mode = false;    //false -> Tocata ; true -> Théremin
int long temps=0;

#define polsador1 2 //Vermell
#define polsador2 3 //Blau

#define analogLEDplay A1            //vermell
#define analogLEDtocata A2          //taronja
#define analogLEDtheremin A3        //blau

//Altaveus
Tone tone1;
Tone tone2;

#define buzzer1 5
#define buzzer2 6

//Registre de desplaçament
#define latchPin 7      // Pin conectat al Pin 12 del 74HC595 (Latch)
#define dataPin 8       // Pin conectat al Pin 14 del 74HC595 (Data)
#define clockPin 12     // Pin conectat al Pin 11 del 74HC595 (Clock)

uint8_t bitsToSend1;
uint8_t bitsToSend2;
uint8_t bitsToSend3;
uint8_t bitsToSend4;
uint8_t bitsToSend5;

//Sensors US
#define triggerPin1 9    // Pin sensor US
#define triggerPin2 13   // Pin sensor US
#define echoPin1 10      // Pin sensor US
#define echoPin2 11      // Pin sensor US
```

```
//Théremin
#define num_mostres 5   // número de mostres que agafa el sensor d'ultrasons
int long distanceCm1;
int long distanceCm2;

int num_nota1ant=0;
int num_nota2ant=0;

//Tocata
int mides[]={(sizeof(melodia11)/sizeof(melodia11[0])),
          (sizeof(melodia21)/sizeof(melodia21[0])),
          (sizeof(melodia31)/sizeof(melodia31[0]))};

const uint16_t *p_melodia1[parts];
const uint16_t *p_melodia2[parts];
const uint16_t *p_duracio[parts];

//Compartit
int num_nota1;
int num_nota2;


void setup() {

  //Compartit
  pinMode(polsador1, INPUT_PULLUP);
  pinMode(polsador2, INPUT_PULLUP);

  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);

  digitalWrite(latchPin, LOW);
  for (int i=0; i<5; i++){
    shiftOut(dataPin, clockPin, LSBFIRST, B00000000);
  }
  digitalWrite(latchPin, HIGH);

  tone1.begin(buzzer1);
  tone2.begin(buzzer2);

  attachInterrupt(0, Interrupcio1, FALLING); //Polsador1
  attachInterrupt(1, Interrupcio2, FALLING); //Polsador2
```

```
  //Tocata
  p_melodia1[0]=&melodia11[0];
  p_melodia1[1]=&melodia21[0];
  p_melodia1[2]=&melodia31[0];

  p_melodia2[0]=&melodia12[0];
  p_melodia2[1]=&melodia22[0];
  p_melodia2[2]=&melodia32[0];

  p_duracio[0]=&duracio1[0];
  p_duracio[1]=&duracio2[0];
  p_duracio[2]=&duracio3[0];
  //Théremin
  pinMode(triggerPin1, OUTPUT);
  pinMode(triggerPin2, OUTPUT);
  pinMode(echoPin1, INPUT);
  pinMode(echoPin2, INPUT);


  //Compartit Final
  analogWrite(analogLEDplay,255);
  analogWrite(analogLEDtocata,0);
  analogWrite(analogLEDtheremin,0);

}


void loop() {

//Pausa

  if (play == false){

    analogWrite(analogLEDplay,255);
    analogWrite(analogLEDtheremin,0);
    analogWrite(analogLEDtocata,0);

    tone1.stop();
    tone2.stop();
    LowPower.powerDown(SLEEP_FOREVER, ADC_OFF, BOD_OFF);
    delay(120);

  }


//Play

  if (play == true){

    analogWrite(analogLEDplay,0);
```

```
//Tocata

  if (mode == false){

    analogWrite(analogLEDtheremin,0);
    analogWrite(analogLEDtocata,255);

    delay(500);

    for (int  i=0;  i<parts; i++){

      if (mode == true){
         tone1.stop();
         tone2.stop();
         break;
      }

      else if (play == false){
         tone1.stop();
         tone2.stop();
         break;
      }

      for (int j=0; j<mides[i]; j++){

        if (mode == true){
          tone1.stop();
          tone2.stop();
          break;
        }

        else if (play == false){
          tone1.stop();
          tone2.stop();
          break;
        }

        num_nota1 = PROGMEM_getAnything(p_melodia1[i]+j);
        num_nota2 = PROGMEM_getAnything(p_melodia2[i]+j);
        int nota1 = PROGMEM_getAnything(&notes[num_nota1]);
        int nota2 = PROGMEM_getAnything(&notes[num_nota2]);
        int duracio = PROGMEM_getAnything(p_duracio[i]+j);


        if (nota2 > 0){
          tone2.play(nota2);
        }
        else{
          tone2.stop();
        }
```

```
if (nota1 > 0){
  tone1.play(nota1);
}else{
  tone1.stop();
}


if (num_nota1==0) {
}
else if ((num_nota1 > 0) && (num_nota1 <= 8)){
  bitWrite(bitsToSend1, (num_nota1-1), HIGH);
}
else if ((num_nota1 > 8) && (num_nota1 <= 16)){
  bitWrite(bitsToSend2, (num_nota1-9), HIGH);
}
else if ((num_nota1 > 16) && (num_nota1 <= 24)){
  bitWrite(bitsToSend3, (num_nota1-17), HIGH);
}
else if ((num_nota1 > 24) && (num_nota1 <= 32)){
  bitWrite(bitsToSend4, (num_nota1-25), HIGH);
}
else if ((num_nota1 > 32) && (num_nota1 <= 40)){
  bitWrite(bitsToSend5, (num_nota1-33), HIGH);
}

if (num_nota2==0)  {
}
else if ((num_nota2 > 0) && (num_nota2 <= 8)){
  bitWrite(bitsToSend1, (num_nota2-1), HIGH);
}
else if ((num_nota2 > 8) && (num_nota2 <= 16)){
  bitWrite(bitsToSend2, (num_nota2-9), HIGH);
}
else if ((num_nota2 > 16) && (num_nota2 <= 24)){
  bitWrite(bitsToSend3, (num_nota2-17), HIGH);
}
else if ((num_nota2 > 24) && (num_nota2 <= 32)){
  bitWrite(bitsToSend4, (num_nota2-25), HIGH);
}
else if ((num_nota2 > 32) && (num_nota2 <= 40)){
  bitWrite(bitsToSend5, (num_nota2-33), HIGH);
}

digitalWrite(latchPin, LOW);
shiftOut(dataPin, clockPin, LSBFIRST, bitsToSend1);
shiftOut(dataPin, clockPin, LSBFIRST, bitsToSend2);
shiftOut(dataPin, clockPin, LSBFIRST, bitsToSend3);
shiftOut(dataPin, clockPin, LSBFIRST, bitsToSend4);
shiftOut(dataPin, clockPin, LSBFIRST, bitsToSend5);
digitalWrite(latchPin, HIGH);
delay(duracio*tempo[i]);
```

```
      if (num_nota1==0) {
      }
      else if ((num_nota1 > 0) && (num_nota1 <= 8)){
        bitWrite(bitsToSend1, (num_nota1-1), LOW);
      }
      else if ((num_nota1 > 8) && (num_nota1 <= 16)){
        bitWrite(bitsToSend2, (num_nota1-9), LOW);
      }
      else if ((num_nota1 > 16) && (num_nota1 <= 24)){
        bitWrite(bitsToSend3, (num_nota1-17), LOW);
      }
      else if ((num_nota1 > 24) && (num_nota1 <= 32)){
        bitWrite(bitsToSend4, (num_nota1-25), LOW);
      }
      else if ((num_nota1 > 32) && (num_nota1 <= 40)){
        bitWrite(bitsToSend5, (num_nota1-33), LOW);
      }

      if (num_nota2==0)  {
      }
      else if ((num_nota2 > 0) && (num_nota2 <= 8)){
        bitWrite(bitsToSend1, (num_nota2-1), LOW);
      }
      else if ((num_nota2 > 8) && (num_nota2 <= 16)){
        bitWrite(bitsToSend2, (num_nota2-9), LOW);
      }
      else if ((num_nota2 > 16) && (num_nota2 <= 24)){
        bitWrite(bitsToSend3, (num_nota2-17), LOW);
      }
      else if ((num_nota2 > 24) && (num_nota2 <= 32)){
        bitWrite(bitsToSend4, (num_nota2-25), LOW);
      }
      else if ((num_nota2 > 32) && (num_nota2 <= 40)){
        bitWrite(bitsToSend5, (num_nota2-33), LOW);
      }

      digitalWrite(latchPin, LOW);
      shiftOut(dataPin, clockPin, LSBFIRST, bitsToSend1);
      shiftOut(dataPin, clockPin, LSBFIRST, bitsToSend2);
      shiftOut(dataPin, clockPin, LSBFIRST, bitsToSend3);
      shiftOut(dataPin, clockPin, LSBFIRST, bitsToSend4);
      shiftOut(dataPin, clockPin, LSBFIRST, bitsToSend5);
      digitalWrite(latchPin, HIGH);

    }
  }
}
```

```
//Théremin

  while (mode == true){

    if (play == false){
      break;
    }

    analogWrite(analogLEDtheremin,255);
    analogWrite(analogLEDtocata,0);

    int long long duration1=0;
    int long long duration2=0;

    for (int i=1; i <= num_mostres; i++){
      digitalWrite(triggerPin1, LOW);
      delayMicroseconds(2);
      digitalWrite(triggerPin1, HIGH);
      delayMicroseconds(10);
      digitalWrite(triggerPin1, LOW);
      duration1 = duration1 + pulseIn(echoPin1, HIGH);
    }

    for (int i=1; i <= num_mostres; i++){
      digitalWrite(triggerPin2, LOW);
      delayMicroseconds(2);
      digitalWrite(triggerPin2, HIGH);
      delayMicroseconds(10);
      digitalWrite(triggerPin2, LOW);
      duration2 = duration2 + pulseIn(echoPin2, HIGH);
    }

    distanceCm1 = duration1 / 58 / num_mostres;
    distanceCm2 = duration2 / 58 / num_mostres;

    if (distanceCm1 >= 60)  {
      num_nota1 = 0;                // LED 0 vol dir tots apagats (silenci)
    }
    else if (distanceCm1 <= 20) {
      num_nota1 = 40;
    }
    else {
      num_nota1 = 60 - distanceCm1;   // objecte a 20 cm -> LED 40 (més agut)
                                      // objecte a 21 cm -> LED 39
                                      // objecte a 59 cm -> LED 1 (més greu)
                                      // objecte a 60 cm -> LED 0 (silenci)
    }
```

```
if (distanceCm2 >= 60)  {
  num_nota2 = 0;
}
else if (distanceCm2 <= 20) {
  num_nota2 = 40;
}
else {
  num_nota2 = 60 - distanceCm2;


}



if (num_nota1ant==0) {
}
else if ((num_nota1ant > 0) && (num_nota1ant <= 8)){
  bitWrite(bitsToSend1, (num_nota1ant-1), LOW);
}
else if ((num_nota1ant > 8) && (num_nota1ant <= 16)){
  bitWrite(bitsToSend2, (num_nota1ant-9), LOW);
}
else if ((num_nota1ant > 16) && (num_nota1ant <= 24)){
  bitWrite(bitsToSend3, (num_nota1ant-17), LOW);
}
else if ((num_nota1ant > 24) && (num_nota1ant <= 32)){
  bitWrite(bitsToSend4, (num_nota1ant-25), LOW);
}
else if ((num_nota1ant > 32) && (num_nota1ant <= 40)){
  bitWrite(bitsToSend5, (num_nota1ant-33), LOW);
}

if (num_nota2ant==0)  {
}
else if ((num_nota2ant > 0) && (num_nota2ant <= 8)){
  bitWrite(bitsToSend1, (num_nota2ant-1), LOW);
}
else if ((num_nota2ant > 8) && (num_nota2ant <= 16)){
  bitWrite(bitsToSend2, (num_nota2ant-9), LOW);
}
else if ((num_nota2ant > 16) && (num_nota2ant <= 24)){
  bitWrite(bitsToSend3, (num_nota2ant-17), LOW);
}
else if ((num_nota2ant > 24) && (num_nota2ant <= 32)){
  bitWrite(bitsToSend4, (num_nota2ant-25), LOW);
}
else if ((num_nota2ant > 32) && (num_nota2ant <= 40)){
  bitWrite(bitsToSend5, (num_nota2ant-33), LOW);
}

delay(180);
```

```
digitalWrite(latchPin, LOW);
shiftOut(dataPin, clockPin, LSBFIRST, bitsToSend1);
shiftOut(dataPin, clockPin, LSBFIRST, bitsToSend2);
shiftOut(dataPin, clockPin, LSBFIRST, bitsToSend3);
shiftOut(dataPin, clockPin, LSBFIRST, bitsToSend4);
shiftOut(dataPin, clockPin, LSBFIRST, bitsToSend5);
digitalWrite(latchPin, HIGH);

int nota1 = PROGMEM_getAnything(&notesTheremin[num_nota1]);
int nota2 = PROGMEM_getAnything(&notesTheremin[num_nota2]);

tone1.stop();
if (num_nota1 > 0)  {
  tone1.play(nota1);
  }
else{
}

tone2.stop();
if (num_nota2 > 0)  {
  tone2.play(nota2);
  }
else{
}

if (num_nota1==0) {
  }
else if ((num_nota1 > 0) && (num_nota1 <= 8)){
  bitWrite(bitsToSend1, (num_nota1-1), HIGH);
}
else if ((num_nota1 > 8) && (num_nota1 <= 16)){
  bitWrite(bitsToSend2, (num_nota1-9), HIGH);
}
else if ((num_nota1 > 16) && (num_nota1 <= 24)){
  bitWrite(bitsToSend3, (num_nota1-17), HIGH);
}
else if ((num_nota1 > 24) && (num_nota1 <= 32)){
  bitWrite(bitsToSend4, (num_nota1-25), HIGH);
}
else if ((num_nota1 > 32) && (num_nota1 <= 40)){
  bitWrite(bitsToSend5, (num_nota1-33), HIGH);
}

if (num_nota2==0)  {
}
else if ((num_nota2 > 0) && (num_nota2 <= 8)){
  bitWrite(bitsToSend1, (num_nota2-1), HIGH);
}
else if ((num_nota2 > 8) && (num_nota2 <= 16)){
  bitWrite(bitsToSend2, (num_nota2-9), HIGH);
}
```

```
      else if ((num_nota2 > 16) && (num_nota2 <= 24)){
        bitWrite(bitsToSend3, (num_nota2-17), HIGH);
      }
      else if ((num_nota2 > 24) && (num_nota2 <= 32)){
        bitWrite(bitsToSend4, (num_nota2-25), HIGH);
      }
      else if ((num_nota2 > 32) && (num_nota2 <= 40)){
        bitWrite(bitsToSend5, (num_nota2-33), HIGH);
      }

      digitalWrite(latchPin, LOW);
      shiftOut(dataPin, clockPin, LSBFIRST, bitsToSend1);
      shiftOut(dataPin, clockPin, LSBFIRST, bitsToSend2);
      shiftOut(dataPin, clockPin, LSBFIRST, bitsToSend3);
      shiftOut(dataPin, clockPin, LSBFIRST, bitsToSend4);
      shiftOut(dataPin, clockPin, LSBFIRST, bitsToSend5);
      digitalWrite(latchPin, HIGH);

      num_nota1ant = num_nota1;
      num_nota2ant = num_nota2;
    }

    bitsToSend1=B00000000;
    bitsToSend2=B00000000;
    bitsToSend3=B00000000;
    bitsToSend4=B00000000;
    bitsToSend5=B00000000;

    digitalWrite(latchPin, LOW);
    for (int i=0; i<5; i++){
      shiftOut(dataPin, clockPin, LSBFIRST, bitsToSend1);
    }
    digitalWrite(latchPin, HIGH);
  }
}

void Interrupcio1() {    //Polsador vermell

  if (millis() > temps + 200){
    play = !play;
    temps=millis();
  }
}

void Interrupcio2() {    //Polsador blau

  if (millis() > temps + 200){
    mode = !mode;
    temps=millis();
  }
}
```

2. Descripció

2.1. Llibreries

A l'inici del programa s'inclouen, amb la instrucció **#include,** totes les llibreries necessàries:

Llibreries públiques

Tone.h[1]: S'utilitza per reproduir dos tons simultàniament en dues sortides digitals diferents de la placa Arduino.

avr/pgmspace.h[2]: S'usa per emmagatzemar dades constants a la memòria flash.

PROGMEM_readAnything.h[3]: Es fa servir per obtenir dades emmagatzemades a la memòria flash.

LowPower.h[4]: S'utilitza per minimitzar el consum d'energia de la placa Arduino quan no està en funcionament.

Llibreries pròpies

Freq.h: Assigna la freqüència corresponent a cada nota.

notesThereminPROGMEM.h: Assigna cada LED de l'orgue a una nota determinada al mode Théremin.

MelodiesTocataPROGMEM.h: Conté les dades de la transcripció de la partitura.

1 https://github.com/bhagman/Tone

2 https://github.com/mikalhart/galileo-Pgmspace.h

3 https://github.com/bcourter/Arduino-Projects/blob/master/tttt/PROGMEM_readAnything.h

4 https://github.com/rocketscream/Low-Power

2.2. Paràmetres i variables

Seguidament es creen les variables i es defineixen els paràmetres que s'utilitzen al llarg del programa. Aquests s'agrupen segons la secció de hardware i mode de l'orgue al que fan referència.

Polsadors i LED dels polsadors

Es creen les variables booleanes **play** i **mode**, una per cada polsador.

També es crea la variable **temps** i se li assigna el valor zero.

A continuació es defineixen, amb el component de C **#define**, els números de les sortides analògiques dels tres LED dels polsadors.

Altaveus

Amb la instrucció **Tone** de la llibreria Tone.h es creen **tone1** i **tone2** per reproduir dos tons independents simultanis.

Després, es defineixen les sortides digitals de la placa Arduino de cada altaveu.

Registre de desplaçament

Es defineixen les sortides digitals de la placa Arduino connectades als xips 74HC595.

Posteriorment, es creen cinc variables (**bitsToSend1**, **bitsToSend2**, etc.), una per cada xip usat en sèrie, per emmagatzemar els bits de dades que es transmetran.

Sensors d'ultrasons

Es defineixen les sortides digitals de la placa Arduino connectades a les potes dels sensors d'ultrasons HC-SR04.

Théremin

En aquest punt, es defineix el paràmetre **num_mostres,** es creen les variables **distanceCm1** i **distanceCm2** i també dues variables amb valor zero, **num_nota1ant** i **num_nota2ant**, que s'utilitzen en la part del programa del mode Théremin.

Tocata

Seguidament, es creen les variables que es fan servir en la part del programa del mode Tocata. Primer es crea un vector anomenat **mides[]** que conté les longituds dels vectors de les tres parts de la partitura.

A continuació, es creen tres vectors de punters buits de longitud tres (una posició per cada part de la partitura). ***p_melodia1[parts]** per la primera melodia, ***p_melodia2[parts]** per la segona melodia i ***p_duracio[parts]** per les duracions.

Compartit

Finalment, es creen les variables enteres **num_nota1** i **num_nota2** que s'utilitzen tant en la part de programa del mode Tocata com en el del mode Théremin.

2.3. Funció setup()

La funció **setup()** està dividida en tres blocs en base al mode de l'orgue fantasmagòric al que fan referència.

Compartit

Primer, es defineixen les potes digitals de la placa Arduino associades als polsadors com a **INPUT_PULLUP** i les potes associades als xips 74HC595 com a sortides.

A continuació, s'envien cinc bits de zeros al registre de desplaçament per apagar tots els LED dels tubs de l'orgue.

Amb la funció **tone1.begin(buzzer1)** es prepara la sortida digital connectada a l'altaveu 1

per reproduir un to. El mateix per l'altaveu 2 amb **tone2.begin(buzzer2)**.

Seguidament es defineixen dues interrupcions, **Interrupcio1** i **Interrupcio2**, una per cada polsador. Aquestes estan associades a les entrades digitals 2 i 3 de la placa Arduino i s'activen quan passen de HIGH a LOW.

Tocata

En aquest punt, s'omplen els vectors de punters ***p_melodia1[parts]**, ***p_melodia2[parts]** i ***p_duracio[parts]** amb la posició de la memòria flash que ocupen les primeres posicions dels vectors que contenen la informació de la partitura de cada part i de les duracions.

Théremin

A continuació, es defineixen com a entrades o sortides digitals les potes digitals de la placa Arduino connectades als sensors d'ultrasons. Les potes connectades a la pota *Trigger* es defineixen com a sortida i les que estan connectades a la pota *Echo* com a entrada.

Compartit final

Finalment, s'encén el LED vermell que fa referència a l'estat de l'orgue fantasmagòric per indicar que està en pausa i que ja pot posar-se en funcionament.

2.4. Funció loop()

La funció loop() està dividida en tres blocs en funció del mode de l'orgue fantasmagòric: mode Pausa, mode Play-Tocata i mode Play-Théremin.

Mode Pausa

Si la variable **play** (que depèn del polsador 1) és *false*, s'il·lumina el LED A1 que indica que l'orgue està en pausa i s'apaguen els LED del mode Tocata i del mode Théremin.

Seguidament, s'atura la possible reproducció de so dels dos altaveus amb **tone1.stop()** i

tone2.stop().

A continuació, la instrucció **LowPower.powerDown(SLEEP_FOREVER, ADC_OFF, BOD_OFF)** adorm la placa Arduino fins que rep una interrupció provinent de qualsevol dels dos polsadors. D'aquesta manera, el consum de la placa disminueix perquè s'eviten iteracions innecessàries quan l'orgue està en pausa.

Mode Play

Si la variable **play** és *true* s'apaga el LED de la sortida A1 per indicar que l'orgue fantasmagòric ja no es troba en pausa.

Finalment, els cinc bits (**bitsToSend1**, **bitsToSend2**...) prenen el valor zero i amb una iteració s'envien al registre de desplaçament per apagar tots els LED.

Mode Tocata

L'ordre que segueix el programa que fa referència al mode Tocata és el següent: obtenir la nota de cada veu i la duració corresponent, reproduir auditivament les notes obtingudes (o aturar el to anterior en cas de silenci), encendre els LED corresponents, esperar la durada de la nota i apagar els LED. A continuació s'exposa com es duen a terme aquestes accions.

Primer de tot, si la variable **mode** és *false*, s'apaga el LED A3 que indica el mode Théremin i s'encén el LED A2 que indica el mode Tocata.

En aquest punt, s'inicia la reproducció de la informació emmagatzemada a les partitures transcrites de l'obra Tocata i Fuga. La reproducció es fa mitjançant dues iteracions: la principal (variable **i**) recorre les tres parts en què està dividida la partitura i la secundària (variable **j**), que està continguda dins de la principal, recorre la partitura de la part corresponent.

A l'inici de la iteració principal hi ha dos **if** que comproven que l'estat de les variables **play** i **mode** sigui el que correspon al mode Tocata. En cas que alguna de les dues variables no sigui la que pertoca, atura la reproducció del so dels dos altaveus i surt de la iteració; si les dues són correctes, es passa a la iteració secundària.

Dins de la iteració que recorre la partitura de la part corresponent, primer de tot es comprova amb dos **if** que l'estat de les variables **mode** i **play** sigui el corresponent. En cas de no ser així, s'atura la reproducció de so en els dos altaveus i surt de la iteració. Si el valor de les variables és l'adequat, comença el procediment per obtenir i reproduir auditiva i

visualment la nota que correspon.

Primer de tot, s'obté el número de LED corresponent (**num_nota1** i **num_nota2**) a partir de la posició de memòria flash que ocupa el primer valor de la partitura de la part corresponent (**i**), a la qual se li suma la variable de la iteració (**j**). De la mateixa manera també s'obté el valor de duració corresponent (**duracio**). El valor de les variables **nota1** i **nota2**, que correspon al nom de la nota de la primera i segona veu, és el valor emmagatzemat a la posició [nota1] i [nota2] del vector **notes[]**.

A continuació, si la freqüència de la variable **nota1** és major de zero, es reprodueix aquesta freqüència per l'altaveu 1. Si la freqüència és zero (silenci), s'atura el so a aquest altaveu. El mateix amb la **nota2**.

El número de LED (**num_nota1** i **num_nota2**) es fracciona en cinc bytes (**bitsToSend1**, **bitsToSend2**...) per ser enviats al registre de desplaçament que consta de cinc xips 74HC595 en sèrie. El bit que correspon al LED se li atorga el valor *HIGH*.

S'envien els cinc bytes al registre de desplaçament i s'aplica el **delay(duracio*tempo[i])**, on el temps d'espera, en mil·lisegons, és el resultat de la multiplicació del valor de duració de la nota corresponent de la iteració secundària pel valor de tempo de la part de la partitura corresponent a la iteració principal.

Un vegada transcorregut el temps del delay(), torna a fraccionar-se el número de LED en cinc bytes. El bit del LED, en aquest cas, se li atorga el valor *LOW*.

S'envien els cinc bytes al registre de desplaçament per tal que s'apaguin els LED que havien sigut encesos. D'aquesta manera, es deixen tots els LED apagats i preparats per la següent iteració. L'altaveu 1 segueix reproduint la **nota1** fins que es reprodueixi una altra freqüència o s'aturi amb un **tone1.stop()** per evitar silencis entre notes. El mateix passa amb la reproducció de la **nota2**.

Mode Théremin

L'ordre que segueix el programa que fa referència al mode Théremin és el següent: obtenir la distància a la qual es troben els obstacles, calcular les notes corresponents, apagar els LED encesos a la iteració anterior, aturar la reproducció dels tons de la iteració anterior, reproduir els tons de les notes actuals i encendre els LED corresponents. D'aquesta manera no hi ha silenci entre notes perquè no s'atura la reproducció auditiva i visual de la nota anterior fins que no s'obté la nota actual. A continuació s'exposa com es duen a terme aquestes accions.

Quan la variable **mode** té valor *true* s'executa el codi del mode Théremin dins d'un *while*. Primer de tot, es comprova si el valor de **play** és *false*; en cas afirmatiu, se surt del *while*.

A continuació, el LED A3 del mode Théremin s'encén i s'apaga el LED A2 del mode Tocata.

Es creen dues variables de valor zero anomenades **duration1** i **duration2** que emmagatzemaran la suma de tots els temps entre polsos obtinguts amb cada sensor d'ultrasons.

S'utilitza una iteració per reproduir el procediment d'obtenció del temps entre polsos tantes vegades com indica el valor **num_mostres**. El temps entre polsos s'obté enviant un pols ultrasònic i contant el temps que transcorre fins que rebota amb l'obstacle i torna al sensor. Quan finalitza la iteració, la variable **duration1** conté la suma del temps entre polsos, en microsegons, obtingut en cada iteració. El mateix amb la segona iteració per obtenir **duration2**.

Seguidament, es converteix la suma de temps entre polsos en la distància a la qual es troba l'obstacle, en centímetres. Aquests valors es guarden a les variables **distanceCm1** i **distanceCm2**.

El rang de distància a la qual treballa el Théremin és de 20 i 60 cm respecte els sensors. A major distància, l'orgue fantasmagòric no reproduirà cap to ni encendrà cap LED. A distàncies inferiors, l'orgue reproduirà la nota més aguda definida i encendrà el LED corresponent. Per tant, si el valor de **distanceCm1** és major de 60 cm, la variable **num_nota1** agafa el valor 0; si **distanceCm1** és inferior de 20 cm, la variable **num_nota1** pren el valor 40, i si **distanceCm1** està entre 20 i 60 cm, **num_nota1** obté un valor igual a seixanta menys el valor de **distanceCm1**. El mateix amb **distanceCm2**.

A continuació, el valor del LED encès a la iteració anterior (**num_nota1ant**) es fracciona en cinc bits (**bitsToSend1**, **bitsToSend2**...) i al LED en qüestió se li atorga el valor *LOW*. El mateix amb **num_nota2ant**.

S'aplica un **delay** de 180 ms perquè els so de l'orgue fantasmagòric en el mode de Théremin sigui clar**.**

S'apaga el LED encès a la iteració anterior (**num_nota1ant**). El mateix amb **num_nota2ant**.

Seguidament, s'obté de la memòria flash els noms de les notes que corresponen als números de LED **num_nota1** i de **num_nota2**, que s'emmagatzemen en les variables **nota1** i **nota2**.

S'atura la reproducció del to de la iteració anterior i, si el valor de **nota1** és major de zero, es reprodueix el to corresponent. El mateix amb **nota2**.

Després, el **num_nota1** es divideix en cinc bits (**bitsToSend1**, **bitsToSend2**...) i al LED corresponent se li atorga el valor *HIGH*. El mateix amb **num_nota2**.

S'envien els cinc bits al registre de desplaçament per il·luminar els LED corresponents.

Finalment, les variables **num_nota1ant** i **num_nota2ant** prenen els valors de **num_nota1** i **num_nota2** respectivament per a la pròxima iteració.

2.5. Interrupcions

Interrupcio1()

La **Interrupció1**, subordinada al polsador de la pota número 2 de la placa Arduino, canvia el valor de la variable booleana **play** cada cop que s'accedeix a la funció, és a dir, quan es polsa el posador i la pota número 2 passa de *HIGH* a *LOW*. Per evitar l'efecte rebot del polsador, la variable **play** només es modifica si han transcorregut més de 200 ms des de l'últim canvi.

Interrupcio2()

La **Interrupció2**, subordinada al polsador de la pota número 3 de la placa Arduino, canvia el valor de la variable booleana **mode** cada cop que s'accedeix a la funció, és a dir, quan es polsa el posador i la pota número 3 passa de *HIGH* a *LOW*. Per evitar l'efecte rebot del polsador, la variable **mode** només es modifica si han transcorregut més de 200 ms des de l'últim canvi.

Annex 2: Llibreries pròpies

1. MelodiesTocataPROGMEM

```
//Melodies Tocata
//*****************************************
#define parts 3
//*****************************************
int tempo[]={66,48,95};
//*****************************************
uint16_t notes[] PROGMEM= {NO,NO,DO1,DOs1,RE1,MI1,FA1,SOL1,LA1,LAs1,
                           DO2,DOs2,RE2,MI2,FA2,FAs2,SOL2,LA2,LAs2,SI2,NO,
                           NO,DO3,DOs3,RE3,MI3,FA3,SOL3,LA3,LAs3,
                           DO4,DOs4,RE4,MI4,FA4,FAs4,SOL4,LA4,LAs4,SI4,NO};

//*****************************************
uint16_t melodia11[] PROGMEM =   {37,36,37,36,34,33,32,31,32,0,
                                  37,36,37,0,33,34,31,32,0,
                                  28,27,28,27,26,25,24,23,24,0,
                                   28,28,28,28,0,23,25,24,24,24,0};

uint16_t melodia12[] PROGMEM =   {17,16,17,16,14,13,12,11,12,0,
                                  17,16,17,0,13,14,11,12,0,
                                  8,7,8,7,6,5,4,3,4,0,
                                  4,11,13,16,18,18,18,16,13,15,0};


uint16_t melodia21[] PROGMEM =   {23,24,25,23,24,25,23,24,25,23,24,25,
                                  26,27,25,26,27,25,26,27,25,26,27,
                                  28,29,27,28,29,27,28,29,27,28,0,
                                  31,32,33,31,32,33,31,32,33,31,32,33,
                                  34,36,33,34,36,33,34,36,33,34,36,
                                  37,38,36,37,38,36,37,38,36,37,0,
                                  37,36,38,33,36,38,33,34,37,32,34,37,32,
                                  33,36,30,33,36,30,32,34,29,32,34,29,
                                  30,33,28,30,33,28,29,32,27,29,32,27,
                                  28,30,26,28,30,26,27,29,25,27,29,25,
                                  26,28,24,26,28,24,25,27,23,25,27,23};

uint16_t melodia22[] PROGMEM =   {11,12,13,11,12,13,11,12,13,11,12,13,
                                  14,16,13,14,16,13,14,16,13,14,16,
```

```
                                17,18,16,17,18,16,17,18,16,17,0,
                                23,24,25,23,24,25,23,24,25,23,24,25,
                                26,27,25,26,27,25,26,27,25,26,27,
                                28,29,27,28,29,27,28,29,27,28,0,
                                28,27,29,25,27,29,25,26,28,24,26,28,24,
                                25,27,22,25,27,22,24,26,18,24,26,18,
                                22,25,17,22,25,17,18,24,16,18,24,16,
                                17,22,14,17,22,14,16,18,13,16,18,13,
                                14,17,12,14,17,12,13,16,11,13,16,11};

uint16_t melodia31[] PROGMEM =  {28,29,28,27,26,25,24,23,19,23,17,23,25,
                                27,27,26,27,26,27,26,27,26,26,25,26,0};

uint16_t melodia32[] PROGMEM =  {4,11,11,11,11,11,11,11,11,11,11,
                                11,11,0,0,0,0,0,0,0,0,0,0,12,0};

uint16_t duracio1[] PROGMEM =   {2,2,20,2,2,2,2,4,12,16,
                                 2,2,12,4,4,4,4,16,16,
                                 2,2,20,2,2,2,2,4,12,16,
                                 4,4,4,4,4,4,24,8,8,16,12};

uint32_t duracio2[] PROGMEM =   {4,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,4,4,
                                2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,4,4,
                                2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,8,12,
                                4,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,4,4,
                                2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,4,4,
                                2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,8,12,
                                4,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,
                                2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,
                                2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,
                                2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,
                                2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6,2.6};

uint16_t duracio3[] PROGMEM = {16,26,2,2,2,2,2,2,2,2,2,2,2,6,2,1,1,1,1,1,1,1,2,4,32,32};
```

2. notesThereminPROGMEM

```
//*** Totes les notes que es reprodueixen amb el Théremin ***//

const uint16_t notesTheremin[] PROGMEM =
        {NO,DO1,DOs1,RE1,REs1,MI1,FA1,FAs1,SOL1,SOLs1,LA1,LAs1,SI1,
        DO2,DOs2,RE2,REs2,MI2,FA2,FAs2,SOL2,SOLs2,LA2,LAs2,SI2,
        DO3,DOs3,RE3,REs3,MI3,FA3,FAs3,SOL3,SOLs3,LA3,LAs3,SI3,
        DO4,DOs4,RE4,REs4};
```

3. Freq

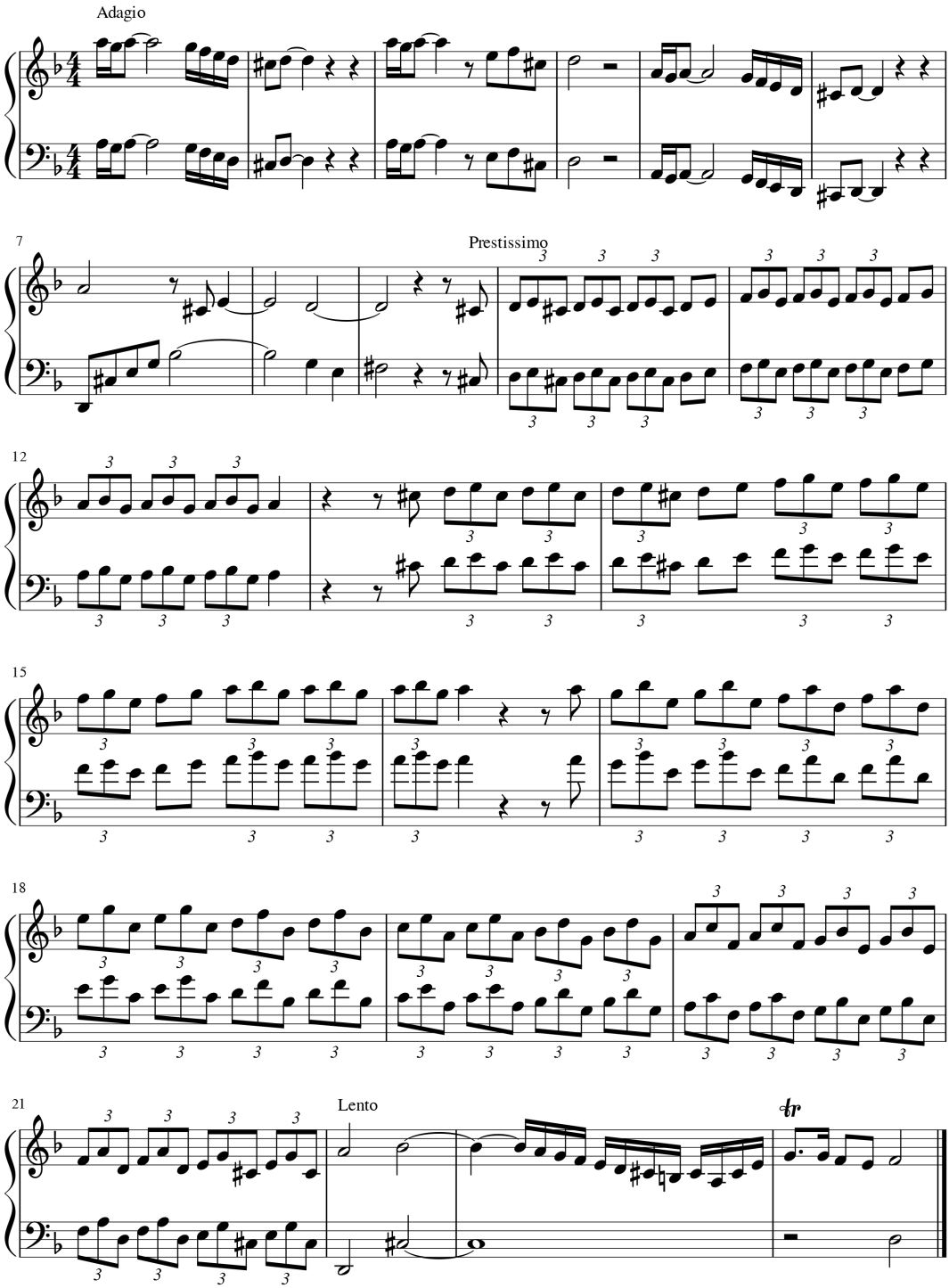
```
//Assignació de la freqüència de cada nota

#define NO      0
#define DO0     33
#define DOs0    35
#define RE0     37
#define REs0    39
#define MI0     41
#define FA0     44
#define FAs0    46
#define SOL0    49
#define SOLs0   52
#define LA0     55
#define LAs0    58
#define SI0     62
#define DO1     65
#define DOs1    69
#define RE1     73
#define REs1    78
#define MI1     82
#define FA1     87
#define FAs1    93
#define SOL1    98
#define SOLs1   104
#define LA1     110
#define LAs1    117
#define SI1     123
#define DO2     131
#define DOs2    139
```

```
#define RE2 147
#define REs2 156
#define MI2 165
#define FA2 175
#define FAs2 185
#define SOL2 196
#define SOLs2 208
#define LA2 220
#define LAs2 233
#define SI2 247
#define DO3 262
#define DOs3 277
#define RE3 294
#define REs3 311
#define MI3 330
#define FA3 349
#define FAs3 370
#define SOL3 392
#define SOLs3 415
#define LA3 440
#define LAs3 466
#define SI3 494
#define DO4 523
#define DOs4 554
#define RE4 587
#define REs4 622
#define MI4 659
#define FA4 698
#define FAs4 740
#define SOL4 784
#define SOLs4 831
#define LA4 880
#define LAs4 932
#define SI4 988
#define DO5 1047
#define DOs5 1109
#define RE5 1175
#define REs5 1245
#define MI5 1319
#define FA5 1397
#define FAs5 1480
#define SOL5 1568
```

```
#define SOLs5   1661
#define LA5     1760
#define LAs5    1865
#define SI5     1976
#define DO6     2093
#define DOs6    2217
#define RE6     2349
#define REs6    2489
#define MI6     2637
#define FA6     2794
#define FAs6    2960
#define SOL6    3136
#define SOLs6   3322
#define LA6     3520
#define LAs6    3729
#define SI6     3951
#define DO7     4186
#define DOs7    4435
#define RE7     4699
#define REs7    4978
```

Annex 3: Partitura adaptada Tocata i fuga



Annex 4: Elements de l'estructura

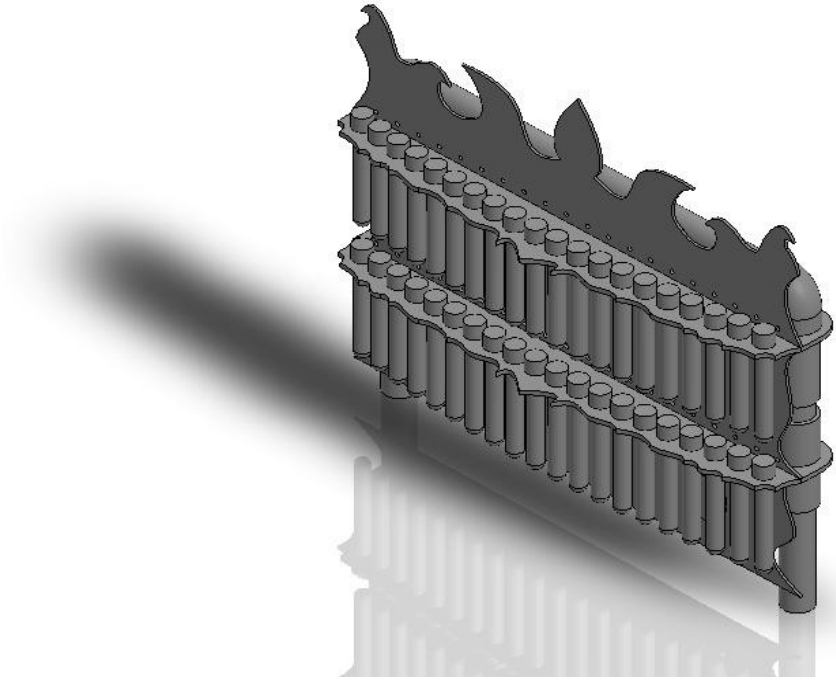


Figura 1: Simulació de l'estructura de l'orgue (Davant)

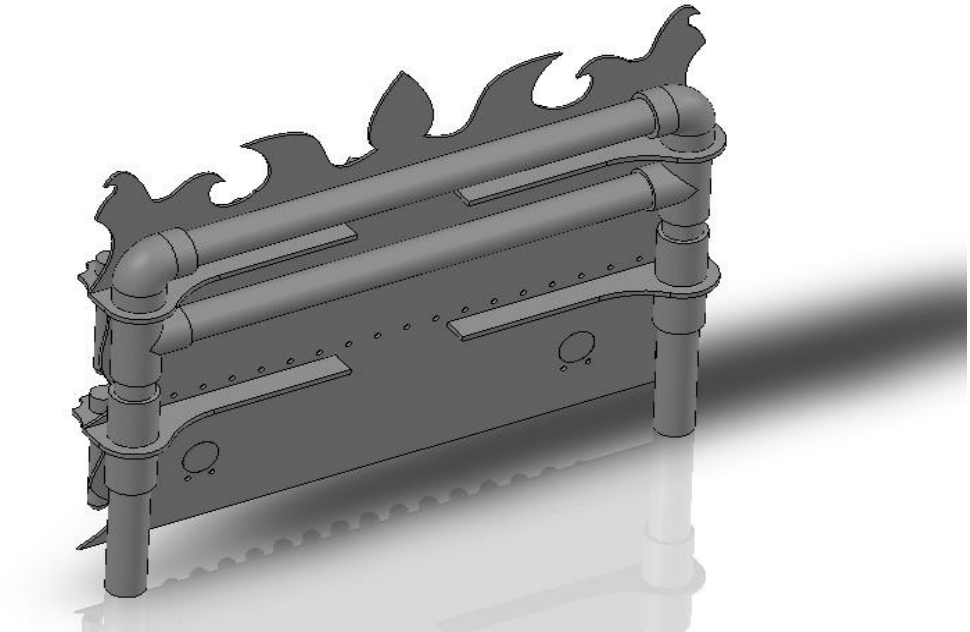


Figura 2: Simulació de l'estructura de l'orgue (Darrera)

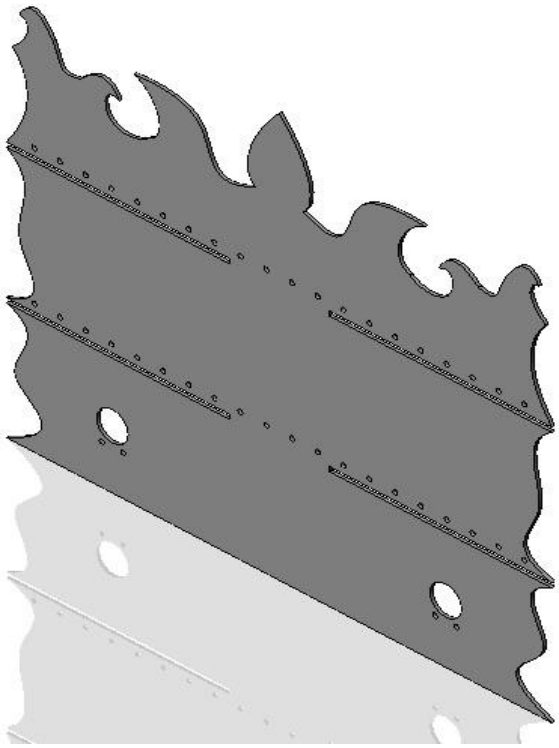


Figura 3: Simulació del fons decoratiu

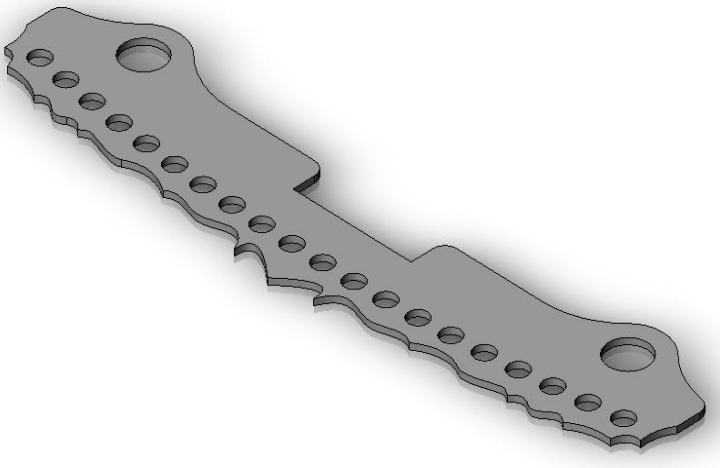


Figura 4: Simulació del suport dels tubs

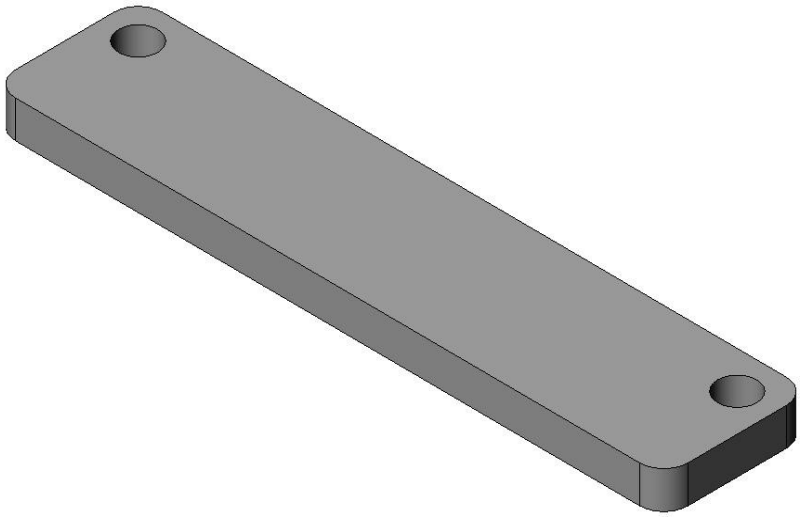


Figura 5: Simulació de la base

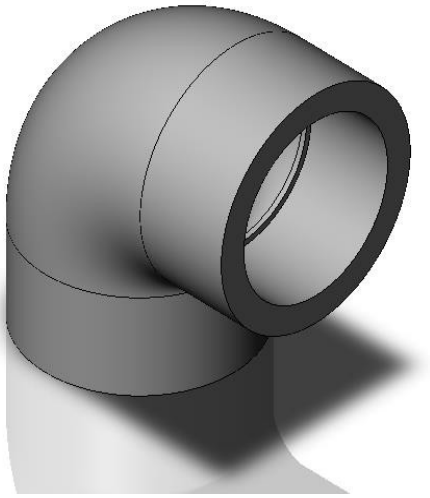


Figura 6: Simulació del colze de 90º

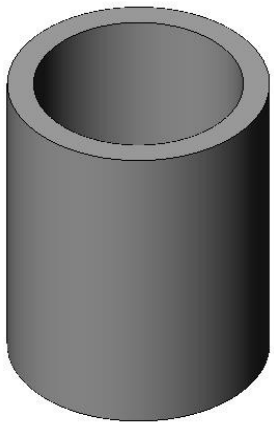


Figura 7: Simulació del maneguet

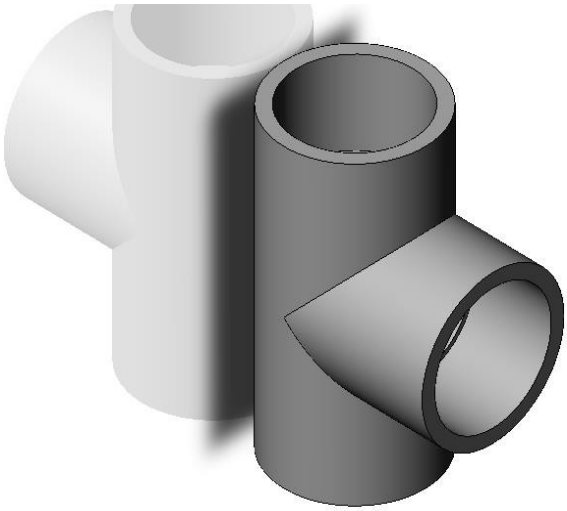


Figura 8: Simulació de la T de 90º

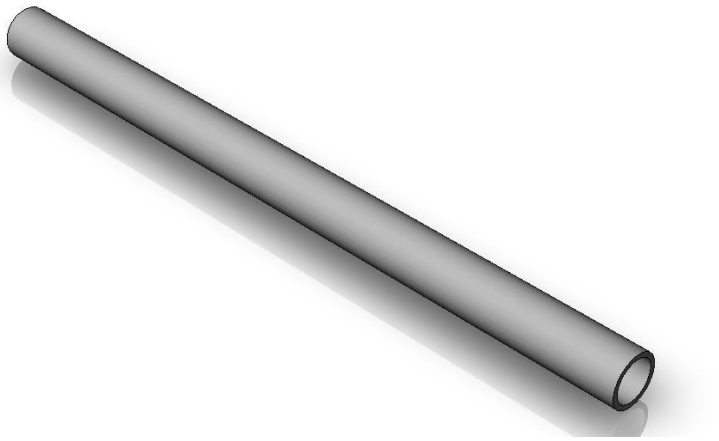


Figura 9: Simulació de la canonada horitzontal inferior

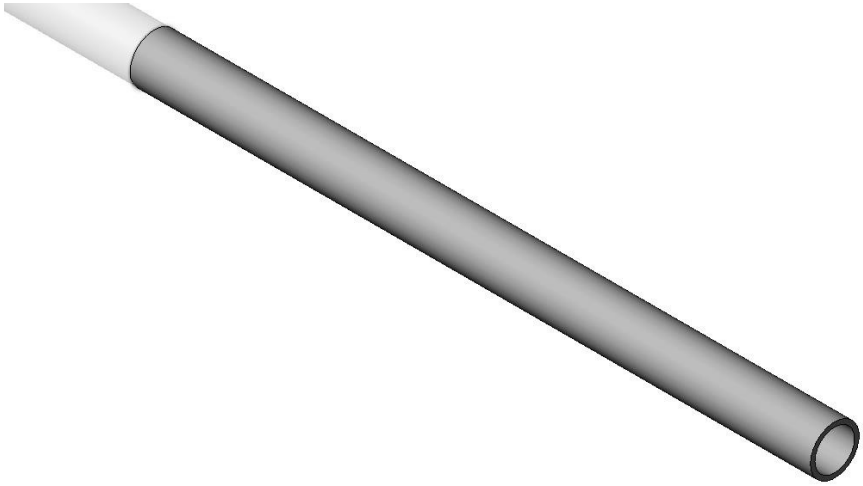


Figura 10: Simulació de la canonada horitzontal superior

Figura 11: Simulació de la canonada vertical inferior

Figura 12: Simulació de la canonada vertical mig-inferior

Figura 13: Simulació de la canonada vertical mig-superior

Figura 14: Simulació de la canonada vertical superior