LAB 06

ISEL – LEIM

11D – Grupo 0

Miguel Alcobia	50746
João Ramos	50730
Daniel Silva	50781

ISEL
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Objetivo do trabalho

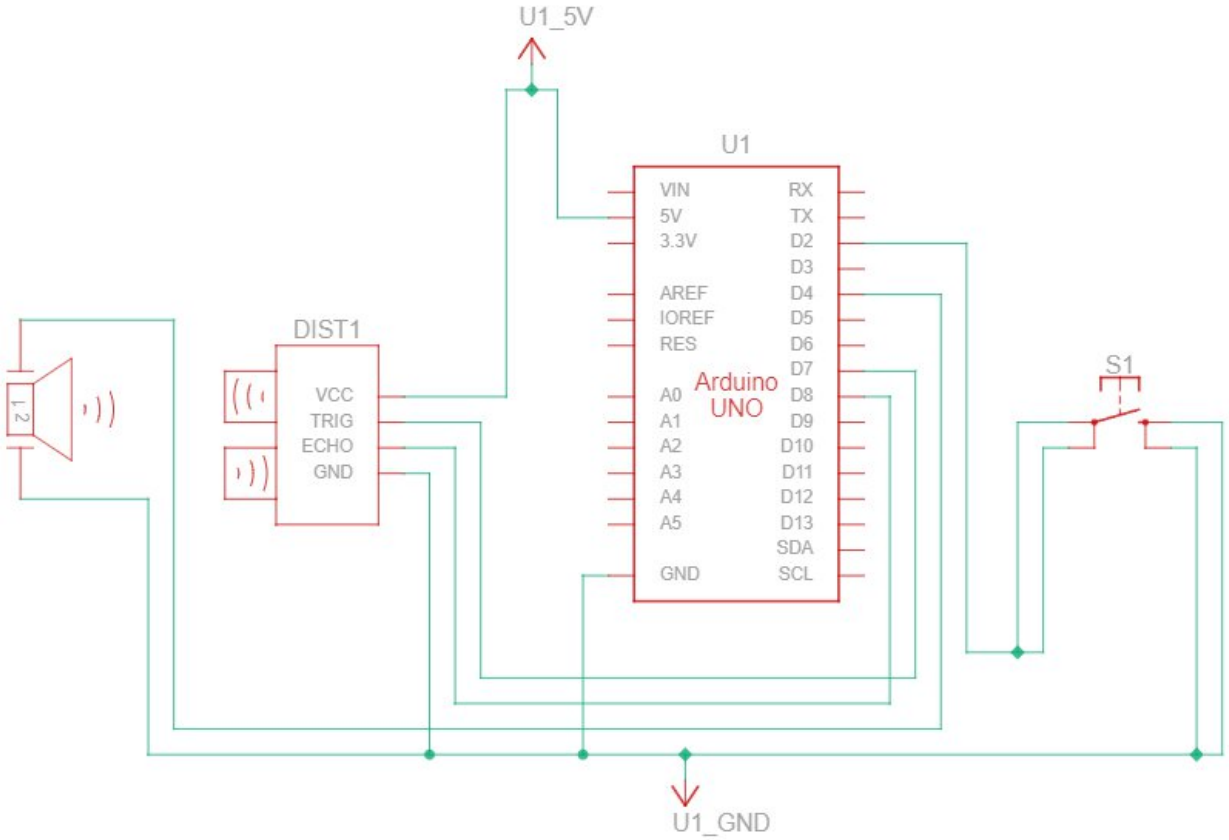
Este trabalho tem como objetivo ensinar os alunos a trabalhar com máquinas de estados e diagramas UML.

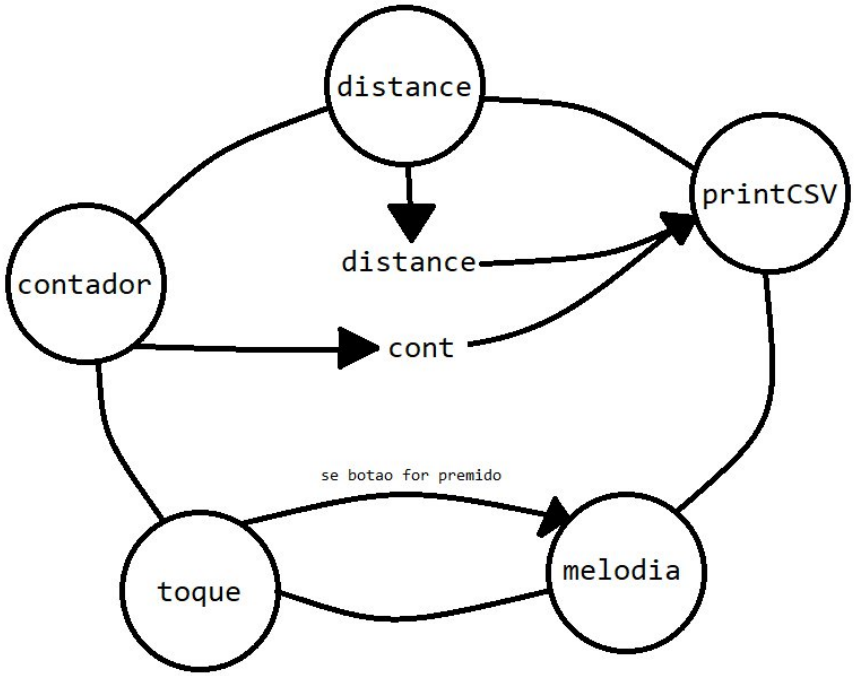
O trabalho consistia em preparar 7 máquinas de estado e depois inseri-las num contexto para integrarem um sistema multi-tarefas.

O contexto inicial escolhido pelo grupo foi uma espécie de alarme onde se o sonar deteta algo a 15cm, o piezo toca um alarme, que pode ser interrompido ao clicar num botão (mesmo ainda dentro do campo alarme delimitado pelo sonar). No entanto a algumas dificuldades com a integração do sonar no código, fez apenas o sonar a trabalhar e o piezo tocava a melodia em loop caso o botão fosse pressionado

Uma impressora CSV envia a distância medida pelo sonar, a quantidade de vezes que o botão foi acionado, e o estado do botão.

Esquema do trabalho





Esquema do trabalho - Continuação

```
#define Trig 7
#define Echo 8
#define S1 2
#define PIEZO 4
#define G2 98
#define A2 110
#define F3 175
bool check;
float dist;
int cont=0;

void setup() {
  Serial.begin(9600);
  pinMode(Trig,OUTPUT);
  pinMode(Echo,INPUT);
  pinMode(S1, INPUT_PULLUP);
  pinMode(PIEZO,OUTPUT);
}
```

```
////////////////////////////////////////
//////SONAR
void distance(int pinTrig, int pinEcho, int
rate){
  //Inicializar os estados
  static const int ESPERA=0;
  static const int ENVIO_PULSO=1;
  static const int ANTES_ECHO=2;
  static const int ECHO_RUN=3;
  static const int MEDE_DISTANCIA=4;

  //inicializar variaveis
  static unsigned long tw=micros();
  static unsigned long t0=millis();
  static unsigned long t1=micros();
  rate = 1000/rate;

  static int state=ESPERA;

  switch (state){
    case ESPERA:
     if (millis()-t0>=rate)
     {
        t0=millis();
        state=ENVIO_PULSO;
     }
    break;
    //Caso ENVIO_PULSO
    case ENVIO_PULSO:
      tw=micros();
      digitalWrite(pinTrig, HIGH);
      if(micros()-tw>=10){
        digitalWrite(pinTrig, LOW);
        tw=micros();
        state=ANTES_ECHO;
      }
    break;
    //Caso ANTES_ECHO
    case ANTES_ECHO:
    if(digitalRead(pinEcho) == HIGH){
      t1=micros();
      state=ECHO_RUN;
      }
    break;
```

```
 / //Caso ECHO_RUN
      case ECHO_RUN:
        if(digitalRead(pinEcho) == LOW){
          state=MEDE_DISTANCIA;
        }
      break;
      //Caso MEDE_DISTANCIA
      case MEDE_DISTANCIA:
        dist=(micros()-t1)/58.0;
        t1=micros();
        t0=millis();
        state=ESPERA;
      break;
    }
 }
/////CSV
void  printCSV(int rate, float val1, float val2,
float val3){
  //Inicializar os estados
  static const int WAIT=0;
  static const int PRINT=1;

  //Inicializar variaveis
  static unsigned long t0=millis(), t1;

  static int state = WAIT;

  switch (state){
    //Caso WAIT
    case WAIT:
      t1=millis()-t0;
      if(t1>1000/rate){
          state = PRINT;
        }
    break;
  //Caso PRINT
    case PRINT:
      Serial.print(val1);
      Serial.print('\t');
      Serial.print('\t'); //derivado ao millis
que ocupa muito espaço
      Serial.print(val2);
      Serial.print('\t');
      Serial.println(val3);
      state = WAIT;
    break;
  }
}
```

```
/////DET
void toque(int pinBotao)
{
  const int PENDING = 0, SCANNING = 1,
PRINTING=2;
  static int state = PENDING;
  static bool b;
  switch (state)
  {
    //Caso PENDING
    case PENDING:
      b = digitalRead(pinBotao);
      if(b == LOW)
      {
        check=false;
        state = SCANNING;
      }
    break;
    //Caso SCANNING
    case SCANNING:
      b = digitalRead(pinBotao);
      if( b == HIGH)
      {
        check=true;
        state = PRINTING;
      }
    break;
    //Caso PRINTING
    case PRINTING:
        Serial.println("Detetado toque no
botão");
        state = PENDING;
    break;
  }
}
```

ISEL INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Esquema do trabalho - Continuação

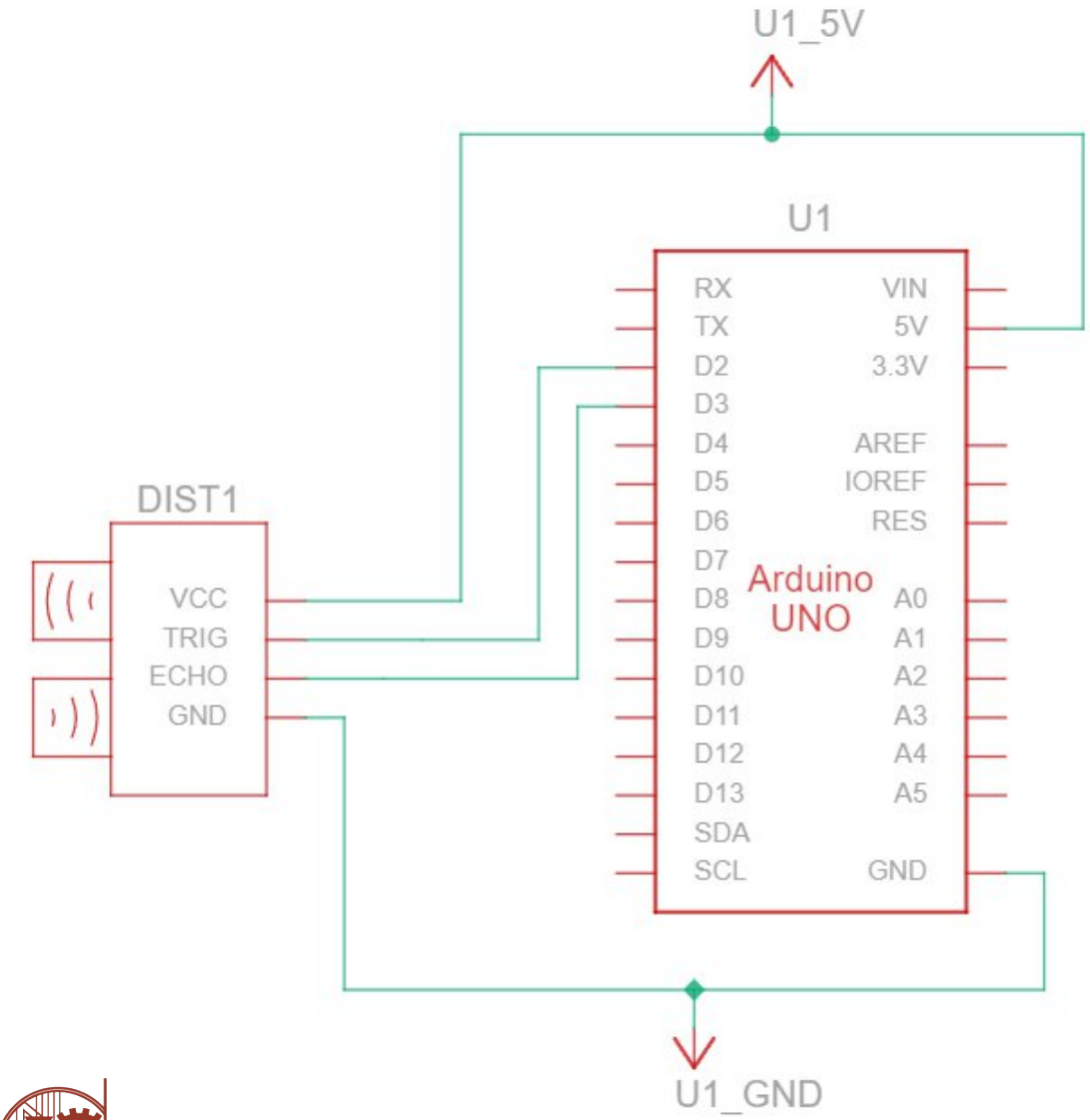
```
/////CONT
void contador(int pin)
{
  const int CONTAGEM = 0 , ESPERA=1, PRINT=2;

  static bool b;

  static int state = CONTAGEM;
  switch (state)
  {
    //Caso CONTAGEM
    case CONTAGEM:
     b = digitalRead(pin);
     if(b==LOW)
     {
       cont= cont + 1;
       state= ESPERA;
     }
    break;
    //Caso ESPERA
    case ESPERA:
     b = digitalRead(pin);
     if (b==HIGH)
     {
       state=PRINT;
     }
    break;
    //Caso PRINT
    case PRINT:
     state=CONTAGEM;
    break;
  }
}}
```

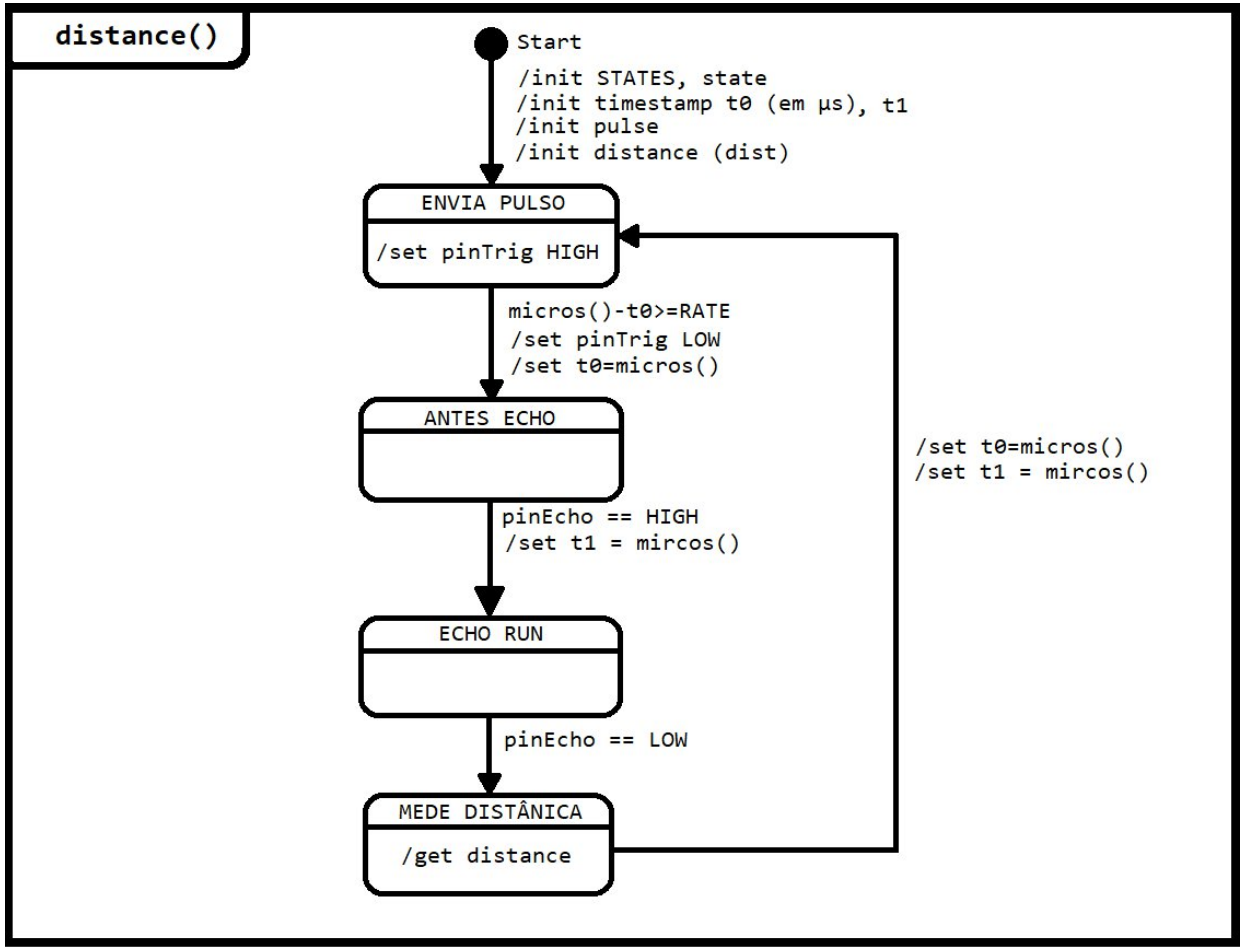
```
/////PIEZO

void melodia(int duration, int nota1, int nota2,
int nota3){
  unsigned long T=duration;
  static const int NOTE_1=0, NOTE_2=1, NOTE_3=2,
ESPERA=3;
  static int state=NOTE_1;
  static unsigned long t0=micros();
  switch (state)
  {
    //Caso NOTE_1
    case NOTE_1:
      if (millis()-t0>T && check==true){
        tone(PIEZO, nota1);
        t0 = millis();
        state = NOTE_2;
      }
    break;
    //Caso NOTE_2
    case NOTE_2:
      if (millis()-t0>T){
        tone(PIEZO, nota3);
        t0 = millis();
        state = NOTE_3;
      }
    break;
    //Caso NOTE_3
    case NOTE_3:
      if (millis()-t0>T){
        tone(PIEZO, nota2);
        t0 = millis();
        state = ESPERA;
      }
    //Caso ESPERA
    case ESPERA:
      if (millis()-t0>T){
        noTone(PIEZO);
        t0 = millis();
        state = NOTE_1;
      }
    break;
 }
}
```

```
void loop() {
  distance(Trig, Echo, 10);
  printCSV(10, millis(), cont, dist);
  melodia(100, G2, A2, F3);
  toque(S1);
  contador(S1);
}
```

Máquinas por Partes

Medidor de distâncias (sonar)



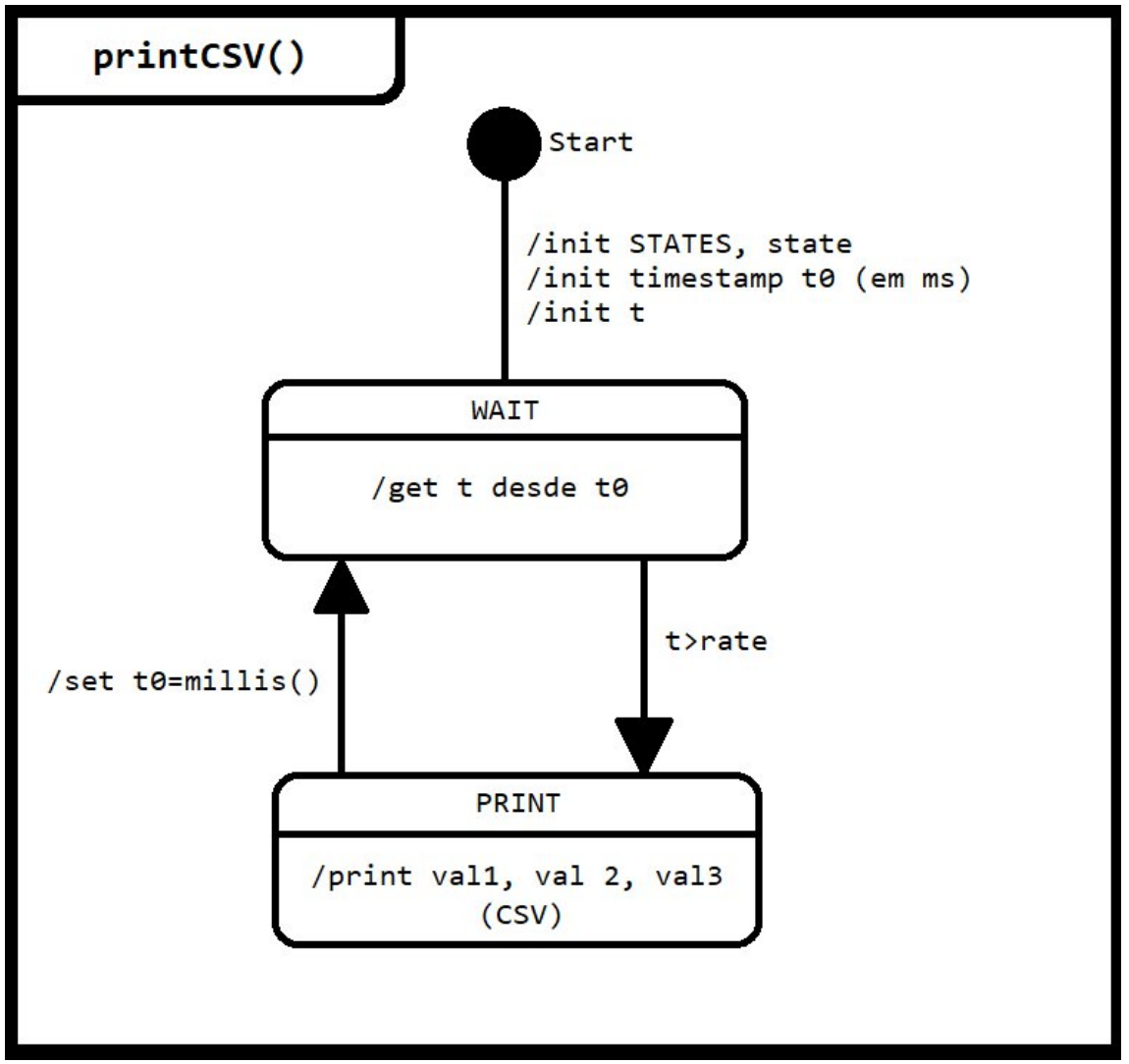


Medidor de distâncias (sonar) - Continuação

```
#define Trig 2
#define Echo 3

void setup() {
  Serial.begin(9600);
  pinMode(Trig,OUTPUT);
  pinMode(Echo,INPUT);
}

void distance(int pinTrig, int pinEcho, int rate){
  static const int ENVIO_PULSO=0;
  static const int ANTES_ECHO=1;
  static const int ECHO_RUN=2;
  static const int MEDE_DISTANCIA=3;

  static unsigned long t0=micros();
  static unsigned long t1=micros();
  float dist;

  static int state=ENVIO_PULSO;

switch (state){
    //Caso ENVIO_PULSO
    case ENVIO_PULSO:
      digitalWrite(pinTrig, HIGH);
      if(micros()-t0>=1000000/rate){
        digitalWrite(pinTrig, LOW);
        t0=micros();
        state=ANTES_ECHO;
      }
    break;
    //Caso ANTES_ECHO
    case ANTES_ECHO:
    if(digitalRead(pinEcho) == HIGH){
      t1=micros();
      state=ECHO_RUN;
    }
    break;
    //Caso ECHO_RUN
    case ECHO_RUN:
      if(digitalRead(pinEcho) == LOW){
        state=MEDE_DISTANCIA;
    }
    break;
    //Caso MEDE_DISTANCIA
    case MEDE_DISTANCIA:
      dist=(micros()-t1)/58.0;
      Serial.println(dist);
      t0=micros();
      t1=micros();
      state=ENVIO_PULSO;
    break;
  }
}

void loop() {
  distance(Trig, Echo, 10);
}
```

Impressora para consola (CSV)

Arduino ligado apenas ao PC

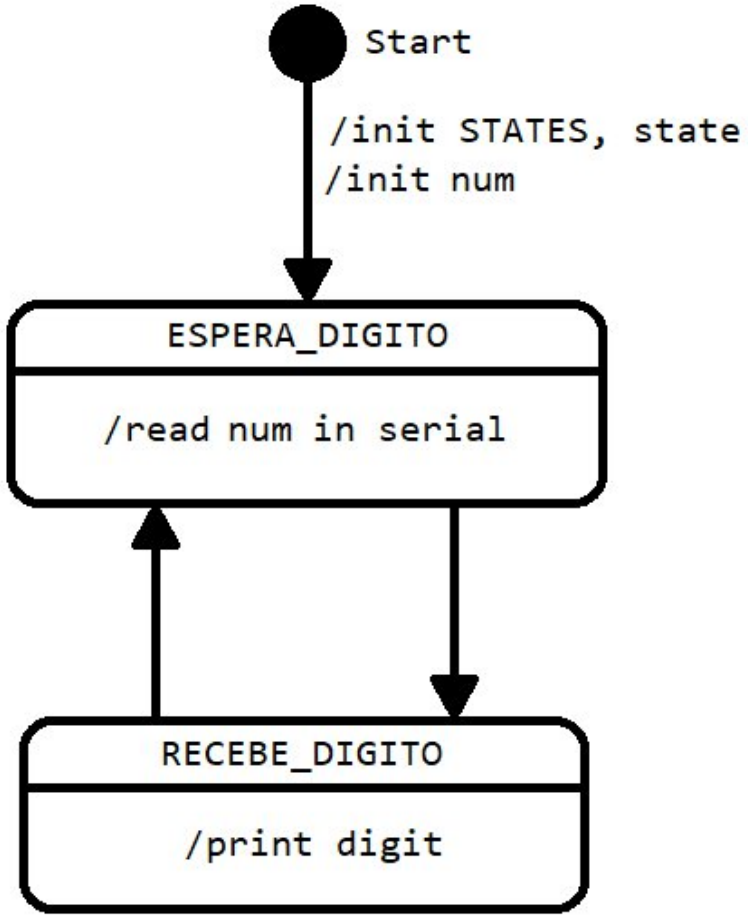


Impressora para consola (CSV) - continuação

```
void setup() {
  Serial.begin(9600);
}

void  printCSV(int rate, float val1, float val2, float
val3){
  static const int WAIT=0;
  static const int PRINT=1;

  static unsigned long t0=millis(), t1;

  static int state = WAIT;

  switch (state){
    case WAIT:
      t1=millis()-t0;
      if(t1>1000/rate){
          state = PRINT;
        }
    break;

    case PRINT:
      Serial.print(val1);
      Serial.print('\t');
      Serial.print(val2);
      Serial.print('\t');
      Serial.println(val3);
      state = WAIT;
    break;
  }
}

void loop() {
  printCSV(2, 2.0, 5.6, 3.2);
}
```

Recetor números inteiros

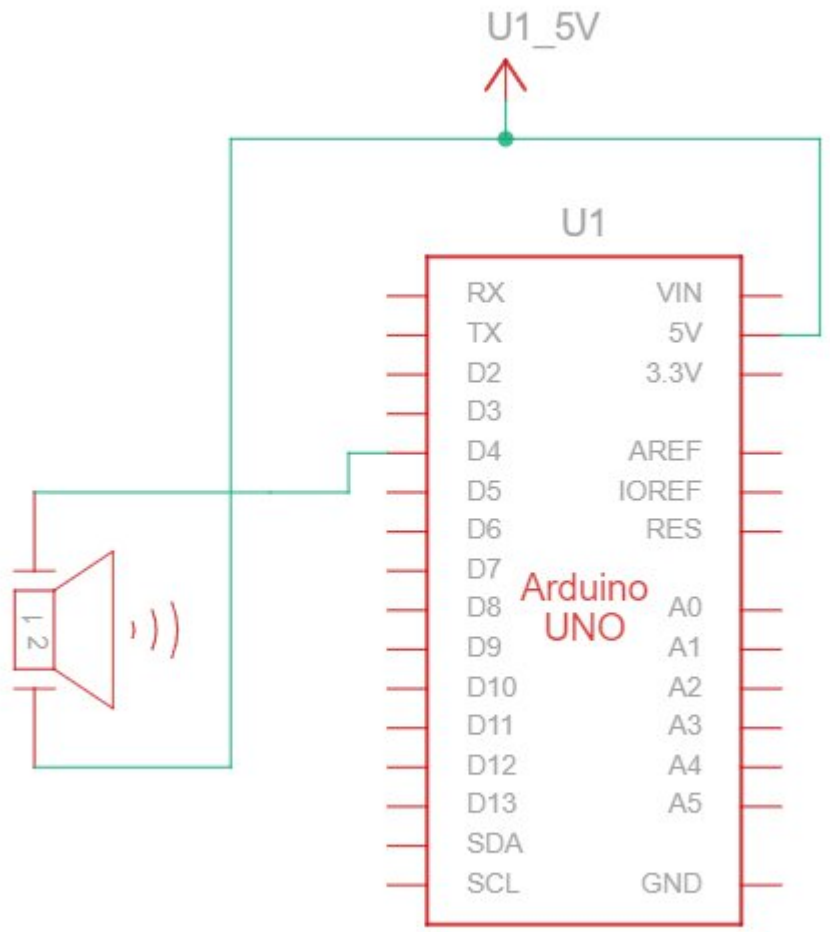
Arduino ligado apenas ao PC

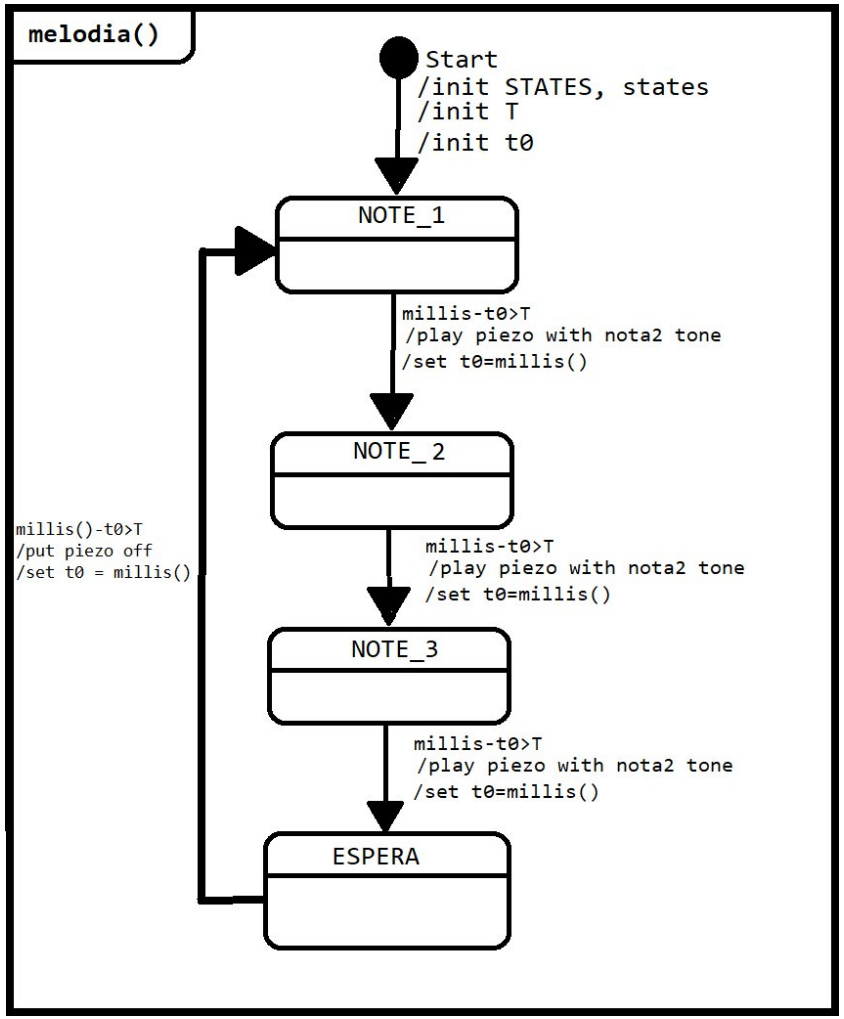


Recetor números inteiros - (continuação)

```
void setup() {
  Serial.begin(9600);
}

void receiveInt(){
  static const int ESPERA_DIGITO=0;
  static const int RECEBE_DIGITO=1;

  static int num=0;

  static int state = ESPERA_DIGITO;

  switch (state){
    case ESPERA_DIGITO:
    if(Serial.available()>0){
          num = Serial.parseInt();
          state = RECEBE__DIGITO;
        }
    break;

    case RECEBE_DIGITO:
        Serial.println(num);
        num = 0;
        state = ESPERA_DIGITO;
  }
}

void loop() {
  receiveInt();
}
```

Tocar melodia (Piezo) - Continuação



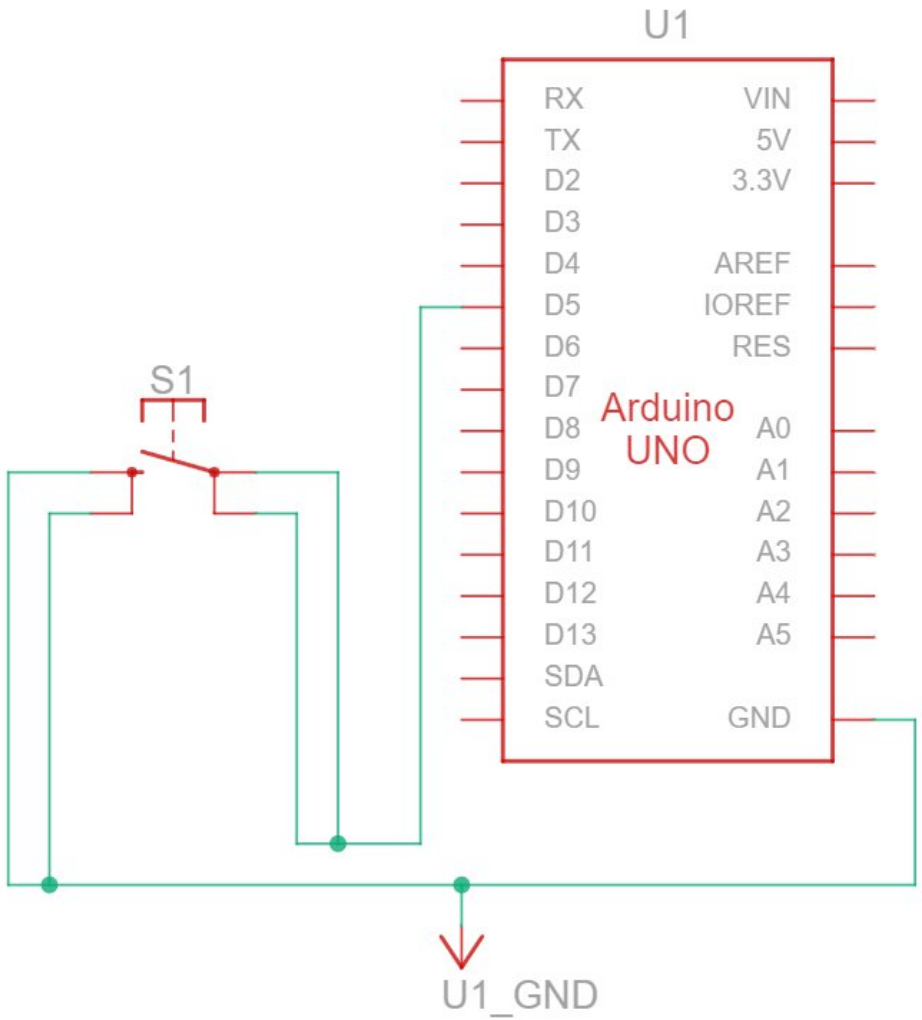


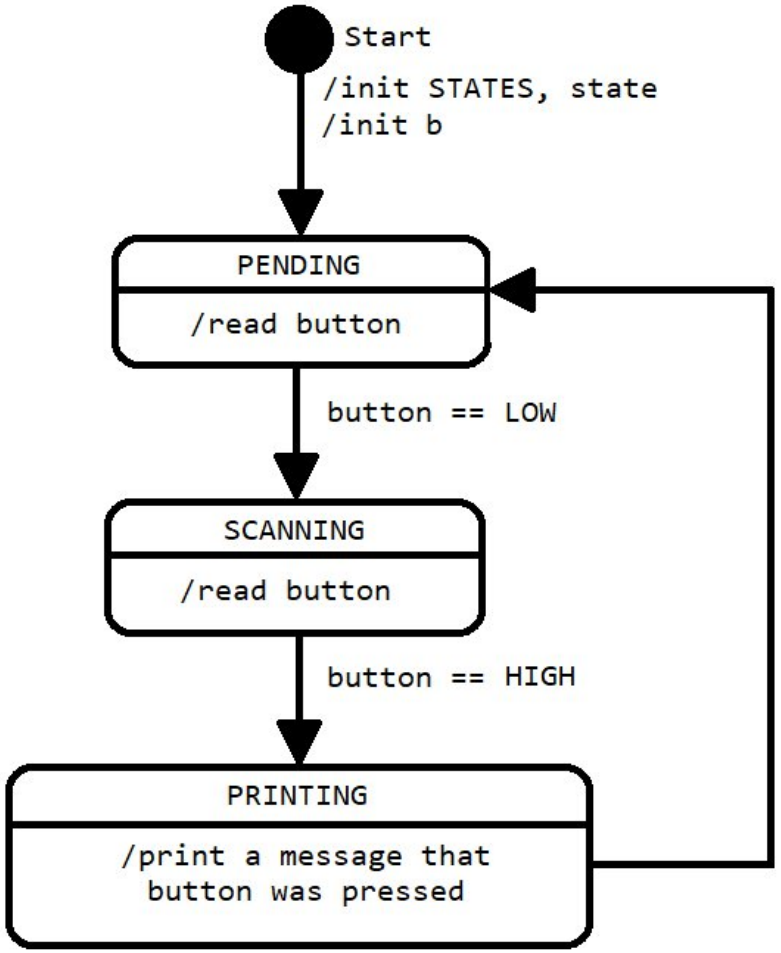
Tocar melodia (Piezo)

```
#define PIEZO 4
#define G2 98
#define A2 110
#define F3 175

void setup()
{
    pinMode(PIEZO,OUTPUT);
}
void loop()
{
  melodia(100, G2, A2, F3);
}

void melodia(int duration, int nota1, int nota2, int
nota3){
  unsigned long T=duration;
  static const int NOTE_1=0, NOTE_2=1, NOTE_3=2,
ESPERA=3;
  static int state=NOTE_1;
  static unsigned long t0=micros();
  switch (state)
  {
    case NOTE_1:
      if (millis()-t0>T){
        tone(PIEZO, nota2);
        t0 = millis();
        state = NOTE_2;
      }
    break;
    case NOTE_2:
      if (millis()-t0>T){
        tone(PIEZO, nota3);
        t0 = millis();
        state = NOTE_3;
      }
    break;
    case NOTE_3:
      if (millis()-t0>T){
        tone(PIEZO, nota1);
        t0 = millis();
        state = ESPERA;
      }
    case ESPERA:
      if (millis()-t0>T){
        noTone(PIEZO);
        t0 = millis();
        state = NOTE_1;
      }
    break;
 }
}
```

ISEL
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Detetor toque (botão)





ISEL
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

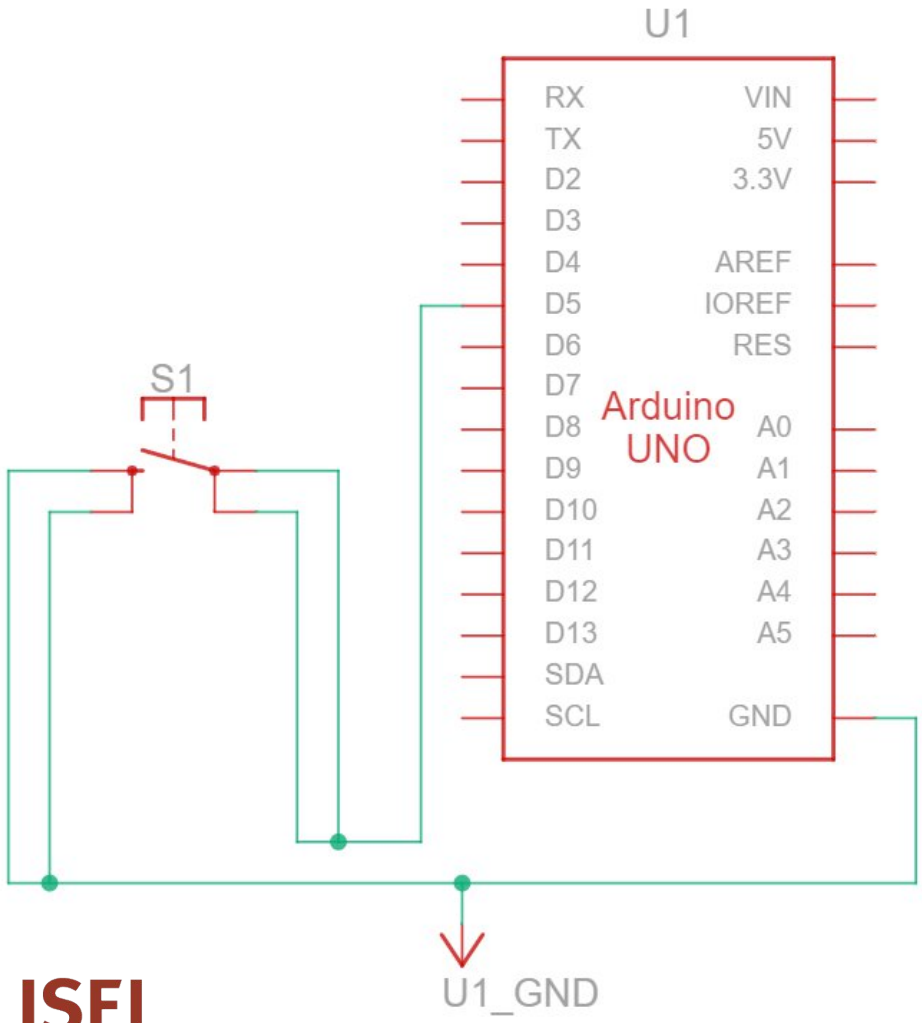
Detetor toque (botão)

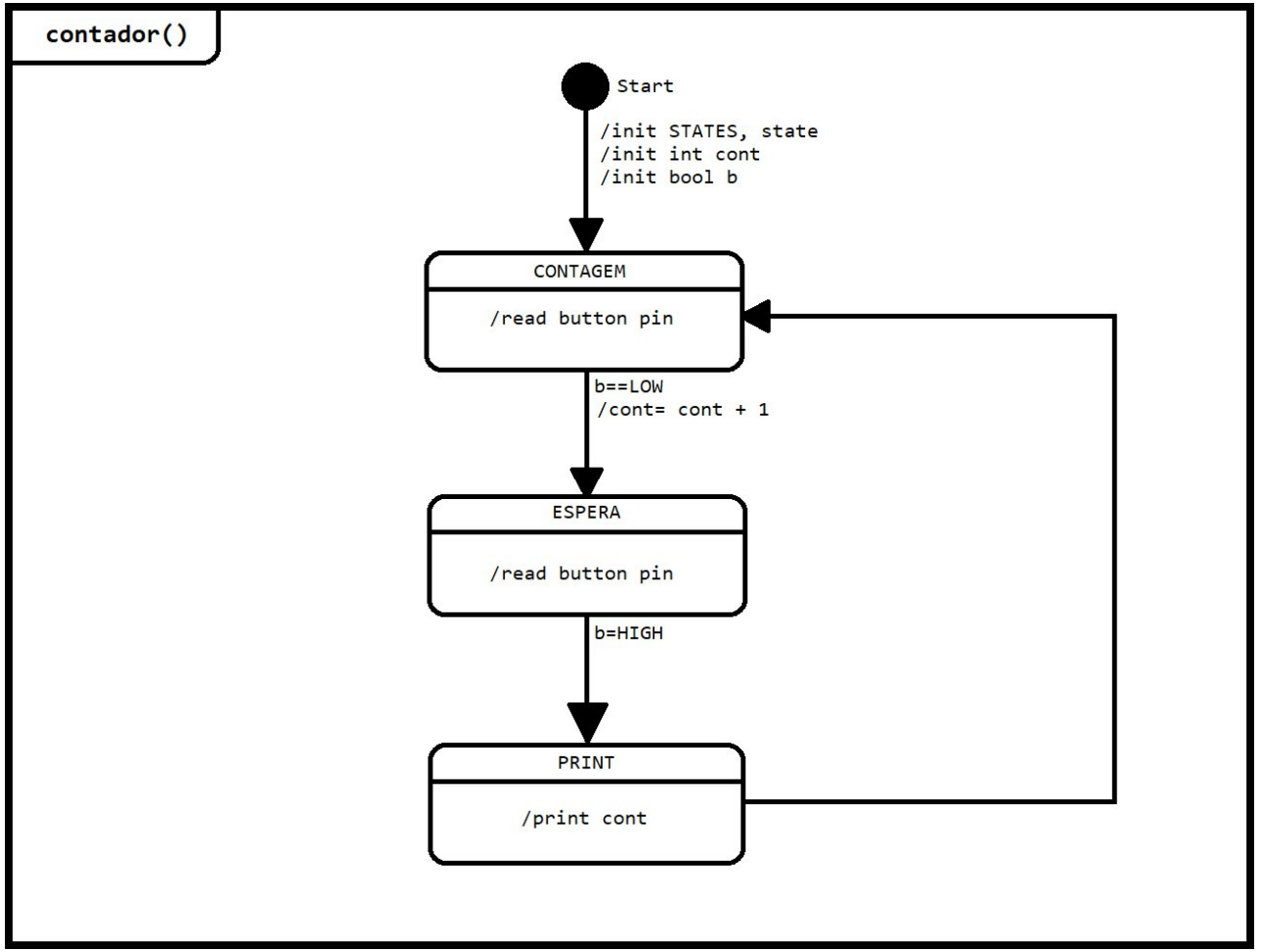
```
#define S1 2
void setup()
{
  Serial.begin(9600);
  pinMode(S1, INPUT_PULLUP);
}

void toque(int pinBotao)
{
  const int PENDING = 0, SCANNING = 1, PRINTING=2;
  static int state = PENDING;
  static bool b;
  switch (state)
  {
    case PENDING:
      b = digitalRead(pinBotao);
      if(b == LOW)
      {
        state = SCANNING;
      }
    break;
    case SCANNING:
      b = digitalRead(pinBotao);
      if( b == HIGH)
      {
        state = PRINTING;
      }
    break;
    case PRINTING:
        Serial.println("Detetado toque no botão");
        state = PENDING;
    break;
  }
}
```

```
void loop()
{
  toque(S1);
}
```

ISEL INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Contador (botão)





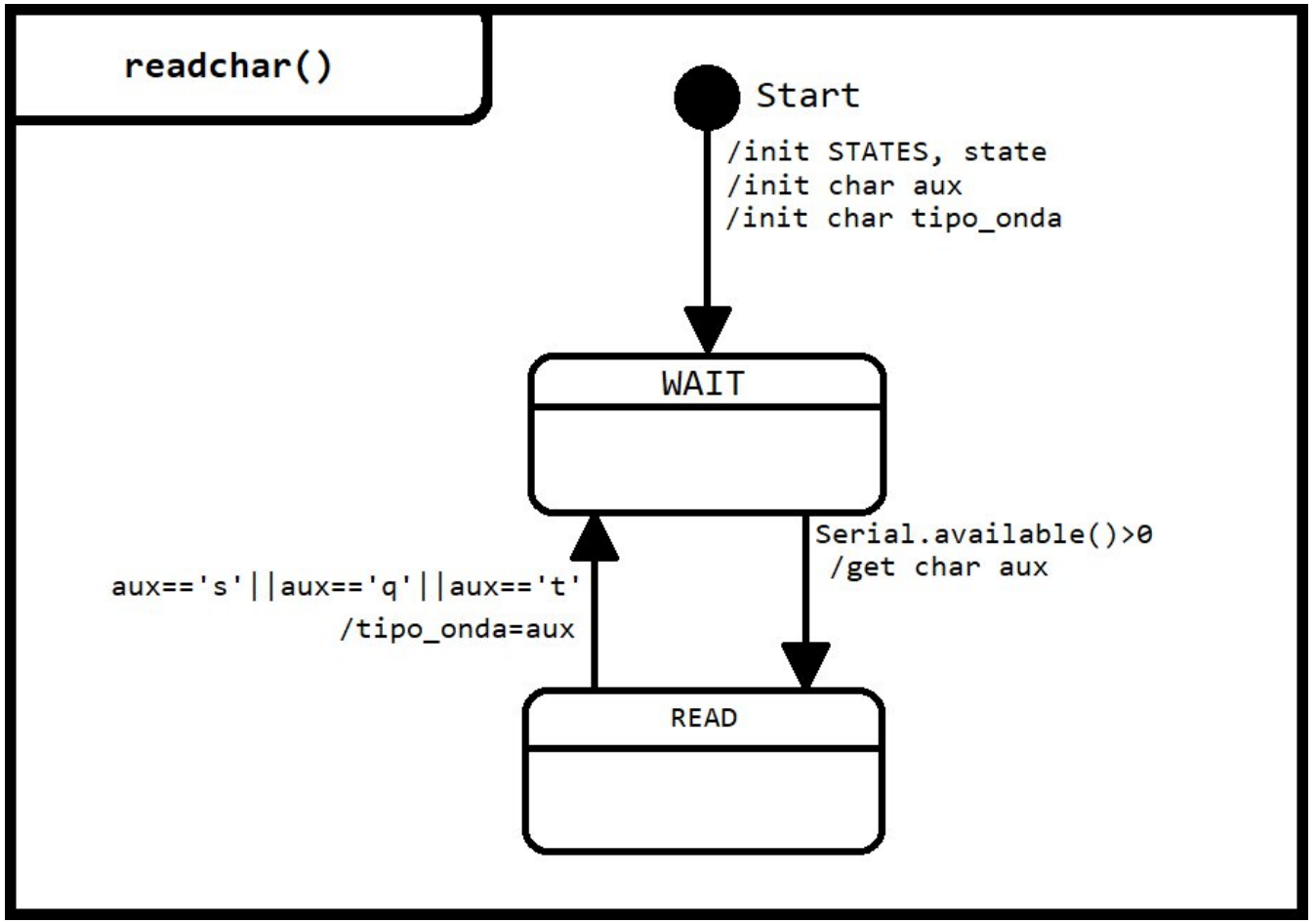
Contador (botão) - Continuação

```
#define S1 2

void setup() {
  Serial.begin(9600);
  pinMode(S1, INPUT_PULLUP);
}

void contador(int pin)
{
  const int CONTAGEM = 0 , ESPERA=1, PRINT=2;

  static int cont;
  static bool b;

  static int state = CONTAGEM;
  switch (state)
  {
    case CONTAGEM:
     b = digitalRead(pin);
     if(b==LOW)
     {
       cont= cont + 1;
       state= ESPERA;
     }
    break;
    case ESPERA:
     b = digitalRead(pin);
     if (b==HIGH)
     {
       state=PRINT;
     }
    break;
    case PRINT:
     Serial.println(cont);
     state=CONTAGEM;
    break;
  }
}


void loop()
{
 contador(S1);
}
```

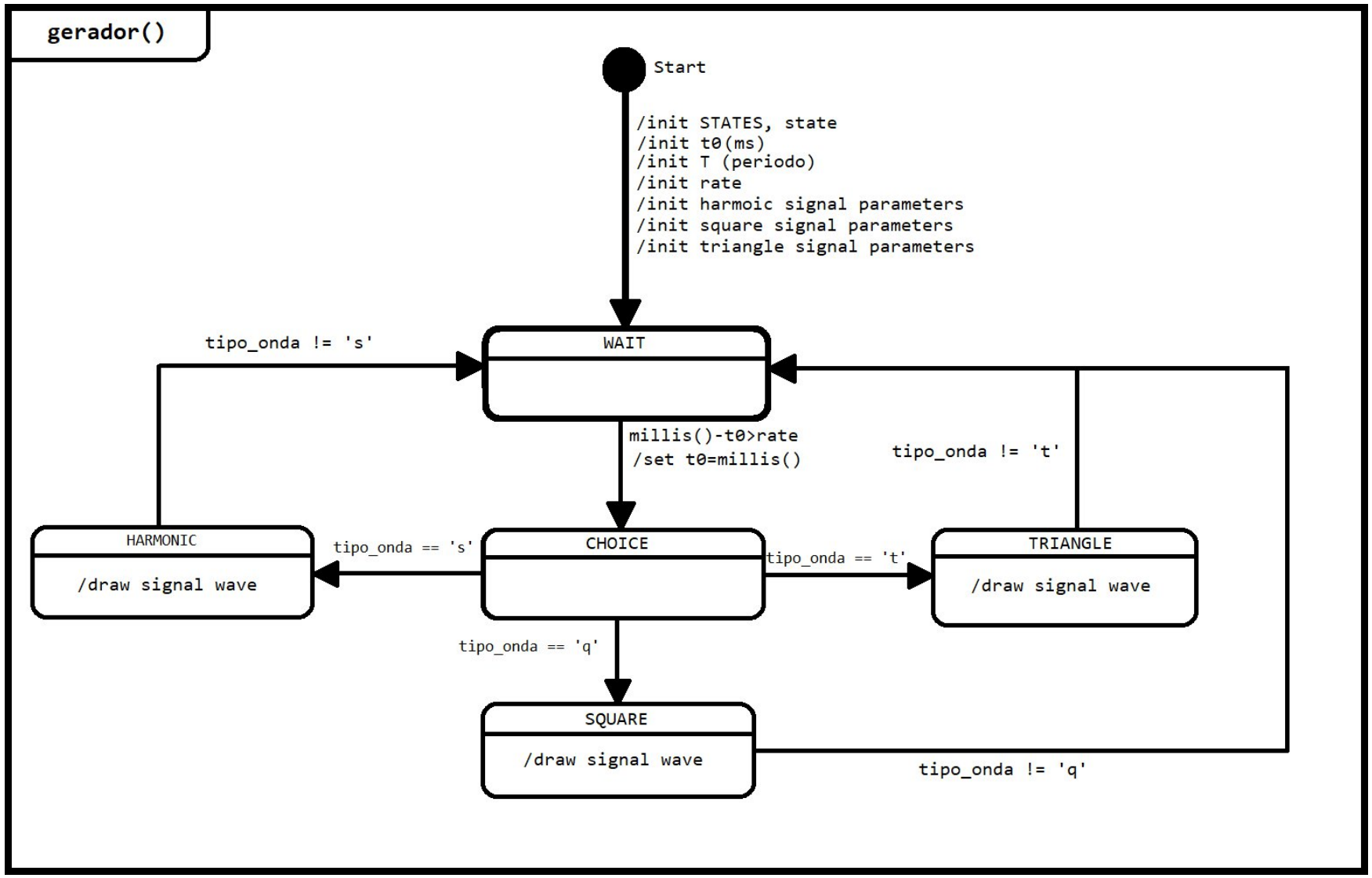
ISEL INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Gerador sinais (onda sinusoidal (s))

Arduino ligado apenas ao PC



Gerador sinais (onda sinusoidal (s)) - Continuação



Gerador sinais (onda sinusoidal (s)) - Continuação

```
char tipo_onda;

void setup()
{
  Serial.begin(9600);
}
void readchar()
{
  static const int WAIT = 0, READ =1;
  static int state = WAIT;
  static char aux;
  switch (state)
  {
    case WAIT:
    if(Serial.available()>0){
      aux = Serial.read();
      state = READ;
    }
    break;
    case READ:
    if(aux=='s'||aux=='q'||aux=='t')
    {
      tipo_onda=aux;
      state=WAIT;
    }
    break;
  }
}
```

```
void  gerador(int rate, int periodo, char tipoOnda){
  static const int WAIT=0, CHOICE=1, HARMONIC=2, SQUARE=3, TRIANGLE=4;
  static unsigned long t0=millis();
  static int state = WAIT;
  int T=periodo;
  rate = 1000/rate;
  //param p/ sinusoidal:
  float ydc=2.5, yac=2.5;//delimitando que o max = 5 e min = 0
  float onda=0;
  int Tst=1000*T/2; //semiperiodo em ms (p/ quadrada e triangular)
  float ymin=0.0,ymax=5.0; //quadrada e triangular
  unsigned long tmin, tmax;

  float ystart,yend;//y value at start/end of semicycle
  float mapyscale=100;
   switch (state)
    {
      case WAIT:
      if(millis()-t0>rate)
      {
        t0=millis();
        state=CHOICE;
      }
      break;
      case CHOICE:
        if(tipo_onda == 's')
          {
            state=HARMONIC;
          }
        if(tipo_onda == 'q')
          {
            state=SQUARE;
          }
        if(tipo_onda == 't')
          {
            state=TRIANGLE;
          }
      break;
      case HARMONIC:
          t0=millis();
        onda = ydc+yac*sin(6.28*t0/1000/T);
        Serial.println(onda);
        if(tipo_onda != 's')
          {
            state=WAIT;
          }
      break;
```

```
case SQUARE: //ymin=0.0,ymax=5.0;; Tst=1000*T/2;
      t0=millis();
        if((t0/Tst)%2==0) Serial.println(ymin);
      else Serial.println(ymax);
      if(tipo_onda != 'q')
        {
          state=WAIT;
        }
      break;
    case TRIANGLE:////ymin=0.0,ymax=5.0;; Tst=1000*T/2;
      tmin=t0/Tst*Tst;   //start time of each semicycle
      tmax=tmin+Tst;  //end time of each semicycle
      if((tmin/Tst)%2==0)
      {
        ystart=ymin;
        yend=ymax;
      }else{
      ystart=ymax;
      yend=ymin;
      }
      t0=millis();
      onda= map(t0,tmin,tmax,mapyscale*ystart,mapyscale*yend)/mapyscale;
      Serial.println(onda);
      if(tipo_onda != 't')
        {
          state=WAIT;
        }
      break;
}
}

void loop()
{
  gerador( 15, 7, tipo_onda);
  readchar();
}
```