Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e Multimédia

LEIM

SENSORES E ATUADORES TRABALHO INDIVIDUAL 3

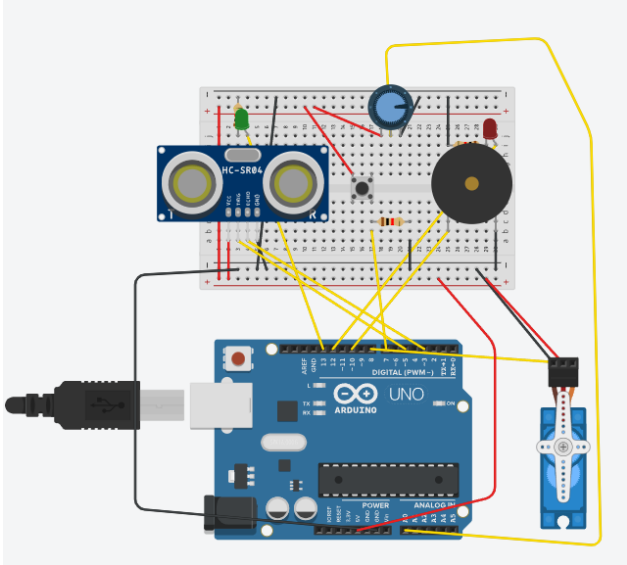
DOCENTES: PROF. MANFRED NIEHUS E PROF ANTÓNIO MAÇARICO

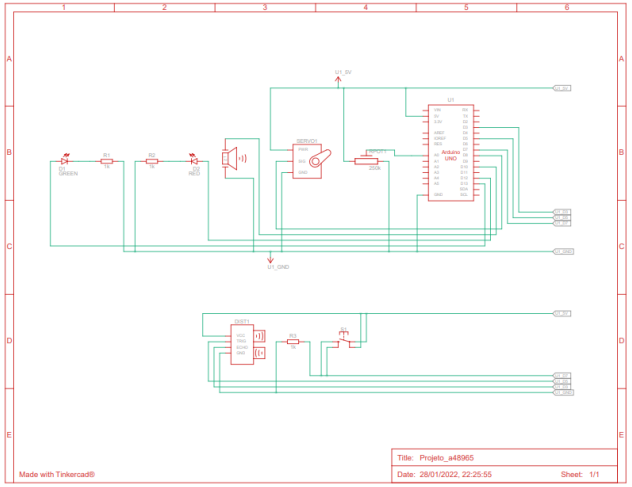
TRABALHO REALIZADO POR: PEDRO SILVA A°48965 TURMA 13D GRUPO B7

CENÁRIO

- O cenário que decidi criar para este projeto é o de um dispensador de comida para animais automático serve refeições de x a x tempo em que é pedido ao animal que se sente num determinado local e que prima um botão para que possa receber a sua comida.

PREPARAÇÃO



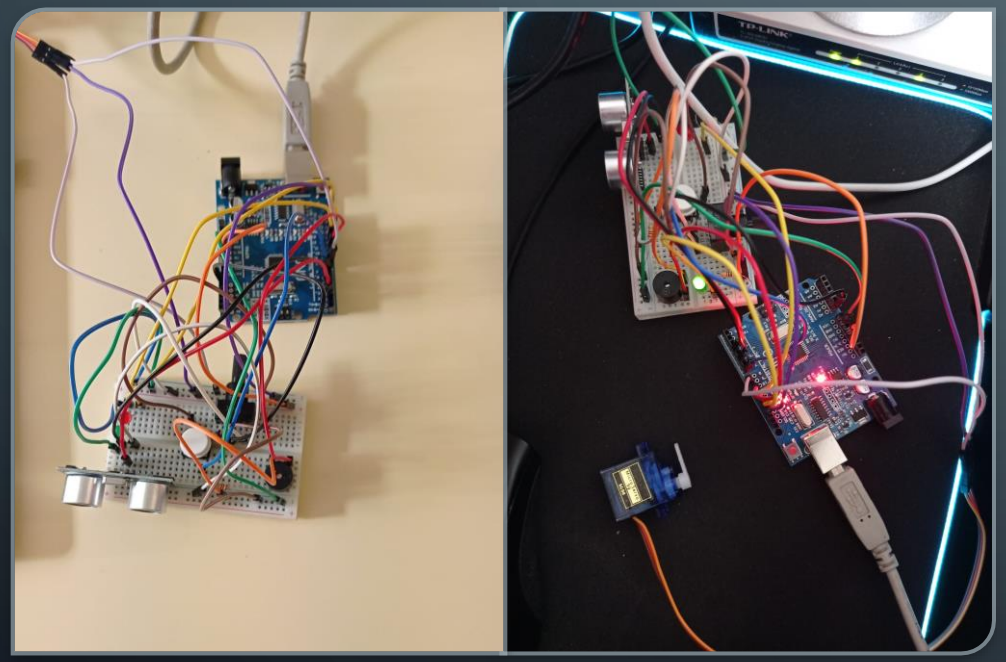


Para fazer a montagem primeiro utilizei o TinkerCad só depois é que o montei fisicamente

Para montar o circuito utilizei:

- 1 Breadboard
- 1 Arduíno
- Jumpers
- 2 LED'S
- 1 Botão
- 3 Resistências
- 1 Motor Servo
- 1 Piezo
- 1 Sonar
- 1 Potenciómetro

MONTAGEM



- Na imagem da esquerda está a primeira montagem e na da direita está a montagem final
- Os LED's mudaram de lugar mas o resto manteve-se

```
#include <Servo.h>

#include <NewPing.h>

//#include para incluir as bibliotecas do sonar e do servo fazendo com que seja possível utilizá-las

#define BALANCA A0
#define echo 3
#define trig 5
NewPing sonar(5, 3);
#define pinServo 8
#define PIEZO 10
#define GREEN 13
#define RED 12
#define BOTAO 7

Servo servo;
bool pesoOK;
bool obstaculoOK=false;
int angle=0;
int reading;
int state;
int aberto=1;
int fechado=0;
bool abre;


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

 servo.attach(pinServo);
 pinMode(BOTAO,INPUT_PULLUP);
 pinMode(PIEZO,OUTPUT);
 pinMode(GREEN,OUTPUT);
 pinMode(RED,OUTPUT);
 //digitalWrite LOW para que ambos os leds começem desligados
 digitalWrite(RED,LOW);
 digitalWrite(GREEN,LOW);
 servo.write(0);


 }


void loop(){

 autoPeso();
 button();
 CANCELA();
}
```

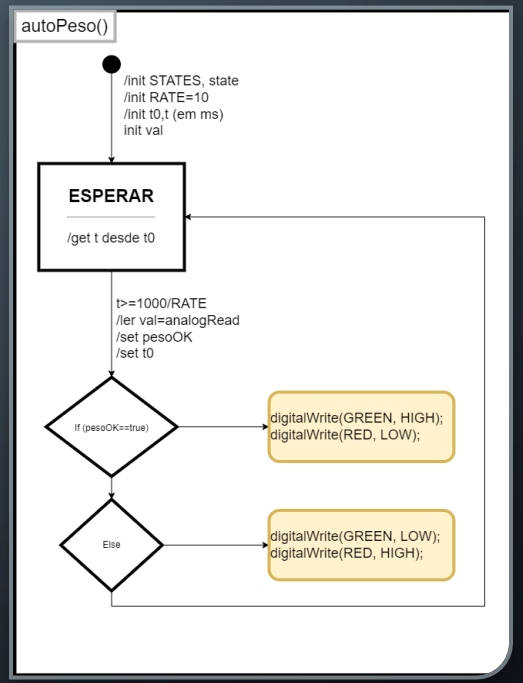
CÓDIGO DO CENÁRIO

- São iniciadas as variáveis que vão ser utilizadas e os pinMode de todos os sensores e atuadores utilizados
- Dentro do void loop dá se o funcionamento do dispositivo

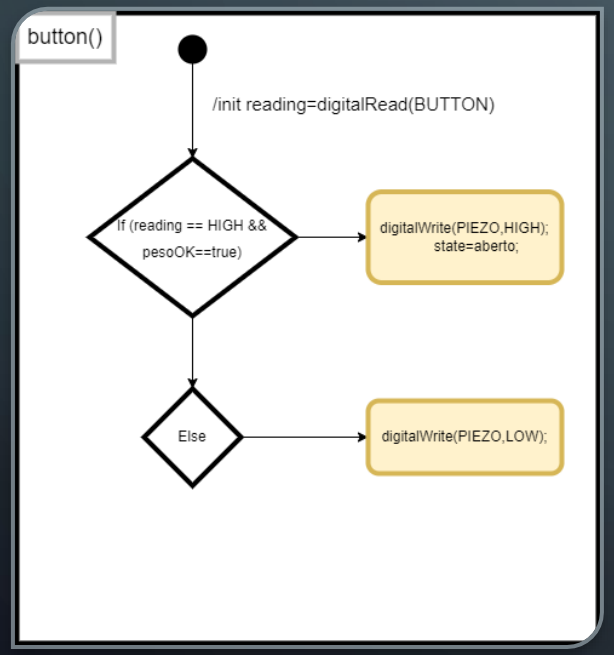
CÓDIGO E UML DO AUTOPESO()

- O automato utilizado está no moodle mas é implementado a mudança de estado dos leds dependendo do pesoOK

```
void autoPeso(){
static const int ESPERAR=0;
static int state=ESPERAR;
static const int RATE=10;
static unsigned long t0=millis();
static unsigned long t;
static int val;
switch(state){
case ESPERAR:
t=millis()-t0;
if(t>=1000/RATE){
val=analogRead(BALANCA);
if(val>102 && val<306)pesoOK=true;
else pesoOK=false;
// se o peso estiver dentro dos requirin
if(pesoOK==true){
     digitalWrite(GREEN, HIGH);
     digitalWrite(RED, LOW);
}else{
     digitalWrite(RED, HIGH);
     digitalWrite(GREEN, LOW);
}
t0=millis();}
break;}}
```



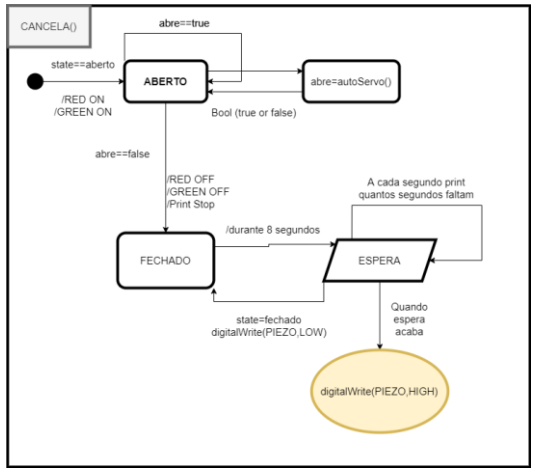
CÓDIGO E UML DO BUTTON()




```
void button(){
 reading=digitalRead(BOTAO);

    if (reading == HIGH && pesoOK==true){
     digitalWrite(PIEZO,HIGH);
     state=aberto;
   }else digitalWrite(PIEZO,LOW);

}
```

- Foi implementada a condição de que se o pesoOK for true e o botão for premido o piezo emite som e o state passa a aberto

CÓDIGO E UML DO CANCELA()

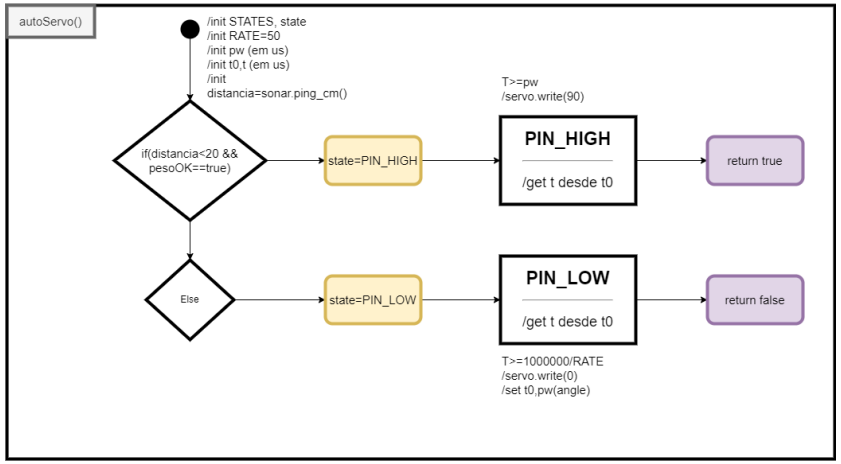



```
void CANCELA(){
 //se o peso estiver dentro dos requirimentos e o estado aberto então ambos os leds acendem e é iniciado o processo de abertura
if(state==aberto ){
     digitalWrite(RED, HIGH);
     digitalWrite(GREEN, HIGH);
     //é atribuido à variável abre true or false.True se o servo estiver aberto (90º) ou false se o servo estiver fechado(0º)
     abre=autoServo();
 //se abre for false vai ser escrito no ecrã serial "Stop" o que significa que o servo fechou e então o state fica fechado também
 if(abre==false){
     digitalWrite(RED, LOW);
     digitalWrite(GREEN, LOW);
     Serial.println("Stop");
     delay(1000);
     //durante 8 segundos vai ser colocado no ecrã serial um temporizador a contar de 8 até 0 que demonstra o tempo de espera até a próxima refeição
     for(int i=8;i>-1;i--){
       Serial.println(i);
       //se a próxima refeição estiver pronta o piezo emite som e é imprimido no Serial "Ready"
       if (i==0){
         Serial.println("Ready");
         digitalWrite(PIEZO,HIGH);
       }
       delay(1000);
   }
     digitalWrite(PIEZO,LOW);
     state=fechado;


 }else state=aberto;
}


}
```

- Quando o state está aberto ambos leds ligam e o servo abre pelo automato autoServo()

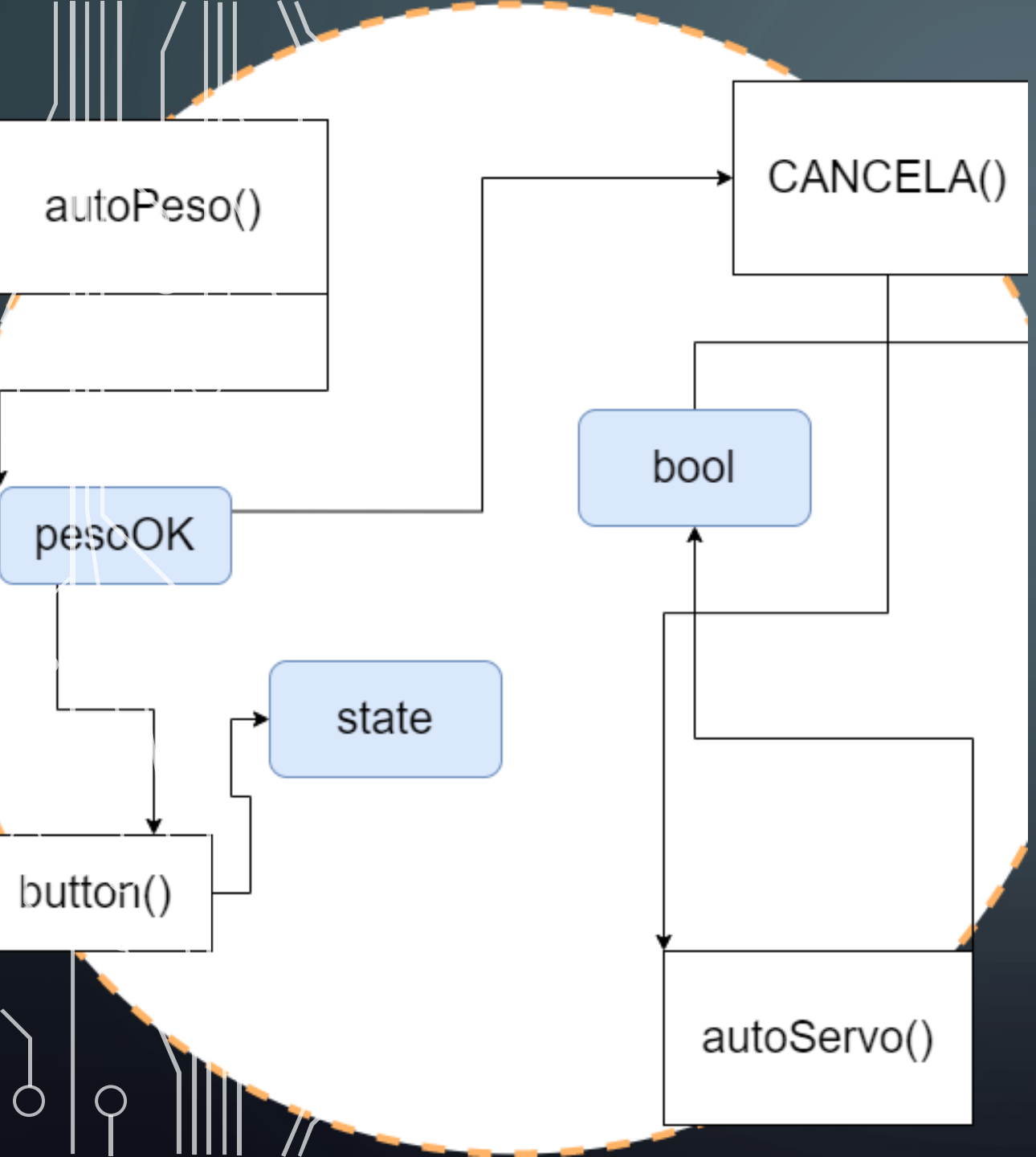
- É utilizado um for para fazer a contagem decrescente até que o dispositivo possa ser utilizado novamente emitindo um som quando esse for o caso

CÓDIGO E UML DO AUTOSERVO()




```
bool autoServo(){
static const int PIN_HIGH=0;
static const int PIN_LOW=1;
static int state=PIN_HIGH;
static const int RATE=50;
static int pw=1500;
static unsigned long t0=micros();
static unsigned long t;
//usar a função sonar.ping_cm() para tranformar o valor do sonar em centimetros
int distancia=sonar.ping_cm();
//se a distancia for menor que 20,ou seja encontrou comida, ou o peso que está na balança deixar de ser true a cancela que liberta a comida fecha se não continua aberta
if(distancia<20 && pesoOK==true){
 state=PIN_HIGH;
}else{
 state=PIN_LOW;
}
switch(state){
case PIN_HIGH:
t=micros()-t0;
if(t>=pw){
servo.write(90);
t0=micros();
return true;
}
break;
case PIN_LOW:
t=micros()-t0;
if(t>=1000000/RATE){
servo.write(0);
t0=micros();
pw=1500+1000/180*angle;
return false;
}
break;}}
```

- Este automato é chamado pelo cancela e devolve um bool
- Se as condições de abertura estiverem bem dá se a abertura da cancela e é devolvido true
- Se estiverem erradas a cancela fecha e é devolvido false



CONCLUSÃO

- Consegui realizar com sucesso o cenário em que tinha pensado, mas devido à falta de tempo não consegui implementar certas funções, como o intervalo de tempo até a próxima refeição e o intervalo de peso fossem determinados pelo utilizador entre outras