

Treball de Fi de Màster

Màster Universitari en Automàtica i Robòtica

Desenvolupament virtual i en context d'un sensor per comandar una plataforma d'ajuda a la mobilitat

ANNEX

Autor/a: Ferran Flaqué Borrellas
Director/a: Joaquim Minguella Canela
Co-director/a: Alicia Casals Gelpí
Convocatòria: Setembre 2023



Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona



Contingut de l'ANNEX

CONTINGUT DE L'ANNEX	3
1. CODIS	4
1.1. Anàlisi del comportament del sensor i filtratge.....	4
1.1.1. Codi per llegir el voltatge de l'Arduino	4
1.1.2. Codi per a llegir Voltatge dels ADC	5
1.2. Codi emprat per la simulació	9
1.2.1. Cadira_descr.....	9
1.2.1.1. URDF	9
1.2.1.1.1 rodaMotor	9
1.2.1.1.2 caster.....	11
1.2.1.1.3 cadira.....	14
1.2.2. Cadira_Control.....	17
1.2.2.1. Config.....	17
1.2.2.1.1 control.....	17
1.2.2.2. Scripts	18
1.2.2.2.1 Main/Main_2.....	18
1.2.2.2.2 Settings.....	22
1.2.2.2.3 Menú.....	24
1.2.2.2.4 LecturaVoltatges	25
1.2.2.2.5 Filtratge.....	27
1.2.2.2.6 ContatgeMinMax	28
1.2.3. Cadira_viz	32
1.2.3.1. Launch	32
1.2.3.1.1 Cadira_vis.....	32

1. Codis

En aquest apartat, es troben els codis utilitzats en la fase d'anàlisi del comportament ja que no han estat comentats en la memòria.

1.1. Anàlisi del comportament del sensor i filtratge

Per a poder fer l'anàlisi dels sensors, s'ha hagut de crear dos codis, un que s'executa sobre l'arduino i que té per finalitat llegir el voltatge i enviar-lo a través del port sèrie i l'altre que s'executa sobre l'ordinador (en format python) i que té per objectiu recollir les dades llegides tant per l'arduino com per l'ADC-10-F103C, i emmagatzemar les dades recollides. Aquest codi, també permet aplicar el filtre de tipus moving average sobre el senyal.

1.1.1. Codi per llegir el voltatge de l'Arduino

El codi emprat per llegir el voltatge de l'Arduino està escrit en una versió adaptada de C:

```
//Codi d'arduino per llegir el voltatge del pont i imprimir-lo en el
port sèrie
#define vDPot A1
#define vGalga A0
float VCC = 4.976;

void setup() {

  cli();
  Serial.begin(9600);
  TCCR1A = 0;
  TCCR1B = 0;
  TCCR1B |= B00000100;
  TIMSK1 |= B00000010;
  TCNT1 = 0;
  OCR1A = 1024;

  sei();

}
void loop() {

}

//Funció per llegir el voltatge
float llegirVoltatgeAmplificat() {
  int Vd = analogRead(vDPot);
  int Vb = analogRead(vGalga);
  float Vamp = (Vd-Vb)*(VCC*(1/1023.0));
  Serial.println(Vamp);
}
```

```
//Interrupció per a llegir el voltatge periòdicament
ISR(TIMER1_COMPA_vect){
    TCNT1 = 0;
    llegirVoltatgeAmplificat();
}
```

1.1.2. Codi per a llegir Voltatge dels ADC

El codi utilitzat per a llegir voltatge dels ADC i emmagatzemar-los ha estat creat emprant python3:

```
import serial
import re
import time
from datetime import datetime

#Definició de la classe filtratge
class Filtratge:
    def __init__(self, Array, Index, Avg):
        self.Array = Array
        self.Index = Index
        self.Avg = Avg
```

```
#Funció per processar l'string rebut per l'ADC-10-F103C
def DemuxData(Data):

    Data_Array = Data.splitlines()

    for x in range(len(Data_Array)):

        Data_String = Data_Array[x].decode("utf-8")
        Data_Vect = re.split(':', Data_String)
        Name = Data_Vect[0]

        if Name == "CH0":
            ADC0 = Data_Vect[1]
            V0 = Data_Vect[2]

        if Name == "CH1":
            ADC1 = Data_Vect[1]
            V1 = Data_Vect[2]

        if Name == "CH2":
            ADC2 = Data_Vect[1]
            V2 = Data_Vect[2]

        if Name == "CH3":
            ADC3 = Data_Vect[1]
            V3 = Data_Vect[2]

        if Name == "CH4":
            ADC4 = Data_Vect[1]
            V4 = Data_Vect[2]

        if Name == "CH5":
            ADC5 = Data_Vect[1]
            V5 = Data_Vect[2]

        if Name == "CH6":
            ADC6 = Data_Vect[1]
            V6 = Data_Vect[2]

        if Name == "CH7":
            ADC7 = Data_Vect[1]
            V7 = Data_Vect[2]

        if Name == "CH8":
            ADC8 = Data_Vect[1]
            V8 = Data_Vect[2]

        if Name == "CH9":
            ADC9 = Data_Vect[1]
            V9 = Data_Vect[2]

    return V0.replace("v", ""), V1.replace("v", "")
#Funció per calcular el voltatge de l'ADC-10-F103C
def llegirVoltatges():

    Data = ser1.read_until(b'\r\n\r\n')

    if (Data):
        Vx = DemuxData(Data)
        vxdac = float(Vx[1]) - float(Vx[0])
    return vxdac
```

```
#Funció per llegir el voltatge de l'arduino
def llegirArduino():
    DataAd = ser2.read_until(b'\r\n')
    DataAdStr = DataAd.decode("utf-8")
    Dvec = DataAdStr.replace("\r\n", "")
    return Dvec

#Funció per aplicar el filtre
def filtrarVoltatge(ValorLlegit, Array, Index):
    Array[Index] = ValorLlegit
    Index = Index + 1
    if (Index >= numLectures):
        Index = 0
    Total = 0.0
    for i in Array:
        Total = Total + i
    Avg = Total/numLectures

    ValorFiltrat = Filtratge(Array, Index, Avg)
    return ValorFiltrat

#Inicialització del port sèrie de l'ADC-10-F103C
Port_ADC = input('Port serie per el ADC?\n')
ser1 = serial.Serial(Port_ADC, 115200)

#Inicialització del port sèrie de l'Arduino per llegir voltatges
Port_Arduino1 = input('Port serie per el Arduino 1?\n')
ser2 = serial.Serial(Port_Arduino1, 9600)

#Inicialització del port sèrie de l'arduino que controla el banc
d'assaig
Port_Arduino2 = input('Port serie per el Arduino del banc?\n')
ser3 = serial.Serial(Port_Arduino2, 9600)

input('Iniciar pruebas?')

#Variables globals emprades
time.sleep(2)
ser2.write(b'llegir')
ser3.write(b'l')
time.sleep(1)
comptador = 0
posicio = 0.0
vdac = 0.0
incrementant = True
stop = False

numLectures = 5
indexLecturesVx = 0
indexLecturesVxArd = 0

arrVx = [0.0 for i in range(numLectures)]
arrVxArd = [0.0 for i in range(numLectures)]
```

```
#Bucle principal del programa
while (not stop):
    #Si rebem missatge de l'ADC-10-F103C, emmagatzamem el valor
    if (ser1.inWaiting()):
        Mat = filtrarVoltatge(llegirVoltatges(), arrVx,
indexLecturesVx)
        vdac = Mat.Avg
        arrVx = Mat.Array
        indexLecturesVx = Mat.Index

    #Esperem a rebre missatge de l'arduino i emmagatzemem el valor (es
produeix cada 0,1s.
    VxArd = llegirArduino()
    Mat2 = filtrarVoltatge(VxArd, arrVxArd, indexLecturesVxArd)
    vard = Mat2.Avg
    arrVxArd = Mat2.Array
    indexLecturesVxArd = Mat2.Index
    print("Timestamp", datetime.now(), end= ";")
    print("Vx = ", vdac, end= ";")
    print("Vard", end= ";")
    print("Posicio = ", posicio)

    #De cada posició es prenen 50 dades, un cop fet s'aplica un nou
increment de posició de 0,1mm
    if (comptador >= 50):
        comptador = 0
        if (incrementant & (not stop)):
            posicio = posicio + 0.1
            ser3.write(b'M 0.1')
        if ((not incrementant) & (not stop)):
            posicio = posicio - 0.1
            ser3.write(b'M -0.1')

    #Un cop s'arriba a la posició màxima, es comença a reduir la
separació en decrements de 0,1mm
    if ((posicio >= 3) & (incrementant)):
        incrementant = False

    if ((posicio < 0.0) & (not incrementant)):
        stop = True
        ser2.write(b'parar')

    comptador = comptador + 1
```

1.2. Codi emprat per la simulació

El codi usat en la simulació s'ha dividit en diferents carpetes amb l'objectiu de mantenir-lo més endreçat.

1.2.1. Cadira_descr

En aquesta carpeta s'hi troba el codi que usat per referenciar la posició de cadascun dels elements mòbils respecte la base (dintre del subdirectori meshes). També s'hi conté els dissenys CAD dels elements (dintre del subdirectori meshes).

1.2.1.1. URDF

És el subdirectori on hi trobem els diferents arxius en format xacro que especifiquen les relacions entre els diferents elements.

1.2.1.1.1 rodaMotor

S'hi conté la macro usada per a referenciar les rodes motores (com que s'ha usat una macro no és necessari crear tants arxius com rodes motores).

```
<?xml version="1.0"?>
<robot name="wheel" xmlns:xacro="http://ros.org/wiki/xacro" >
<!-- inicialització de variables -->

  <xacro:property name="roda_kp" value="600000" />
  <xacro:property name="roda_kd" value="3" />
  <xacro:property name="roda_mu1" value="10000000" />
  <xacro:property name="roda_mu2" value="10000000" />
  <xacro:property name="M_PI" value="3.14" />
```

```

<!--Definició de la macro de la roda motora -->
<xacro:macro name="wheel" params="wheel_prefix parent_link *joint_pose
*axis">

  <!-- Definició de l'eslabó de la roda motora -->
  <link name="${wheel_prefix}_wheel_link">
  <inertial>
    <origin xyz="0 0 0"/>
    <mass value="0.50000000"/>
    <inertia ixx="0.00276879" ixy="-0.00000043" ixz="-
0.000000140" iyy="0.00520797" iyz="0.00000030" izz="0.00276862"/>
  </inertial>
  <visual>
    <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0" />
    <geometry>
      <mesh
filename="package://cadira_description/meshes/Roda_Motor.dae" />
    </geometry>
  </visual>
  <collision>
    <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0" />
    <geometry>
      <mesh
filename="package://cadira_description/meshes/Roda_Motor.dae" />
    </geometry>
  </collision>
</link>

  <!-- Definició de l'articulació de la roda motora -->
  <joint name="${wheel_prefix}_wheel" type="continuous">
    <parent link="${parent_link}"/>
    <child link="${wheel_prefix}_wheel_link"/>
    <xacro:insert_block name="joint_pose"/>
    <xacro:insert_block name="axis"/>
    <dynamics damping="0.5" friction="100.0" />
  </joint>

  <!-- Codi específic per a gazebo -->
  <gazebo reference="${wheel_prefix}_wheel_link">
    <mul value="${roda_mu1}"/>
    <mu2 value="${roda_mu2}"/>
    <kp>${roda_kp}</kp>
    <kd>${roda_kd}</kd>
    <dampingFactor>0.05</dampingFactor>
    <fdirl value="1 0 0"/>
  </gazebo>

```

```

    <!--Relacions entre actuadors i articulacions -->
    <transmission name="{wheel_prefix}_wheel_trans"
type="SimpleTransmission">
        <type>transmission_interface/SimpleTransmission</type>
        <actuator name="{wheel_prefix}_wheel_motor">
            <mechanicalReduction>1</mechanicalReduction>
        </actuator>
        <joint name="{wheel_prefix}_wheel">

<hardwareInterface>hardware_interface/VelocityJointInterface</hardware
Interface>
        </joint>
    </transmission>

</xacro:macro>

</robot>

```

1.2.1.1.2 caster

S'hi combina tant la base de la roda Caster com la roda Caster en una mateixa macro.

```

<?xml version="1.0"?>
<!-- inicialització de variables -->
<robot xmlns:xacro="http://ros.org/wiki/xacro" >

    <xacro:property name="wheel_kp" value="600000" />
    <xacro:property name="wheel_kd" value="3" />
    <xacro:property name="wheel_mu1" value="10000000" />
    <xacro:property name="wheel_mu2" value="10000000" />

    <!--Definició de la macro de la roda Caster -->
    <xacro:macro name="caster" params="caster_prefix parent_link
*joint_pose">

        <!-- Definició de l'eslabó de la roda Caster -->
        <link name="{caster_prefix}_caster_link">
            <inertial>
                <origin xyz="0 0 0"/>
                <mass value="0.10000000"/>
                <inertia ixx="0.00018511" ixy="-0.00000000" ixz="-
0.00000000" iyy="0.00031795" iyz="0.00000000" izz="0.00018511"/>
            </inertial>

```

```

    <visual>
      <origin xyz="0 0 0" rpy="0 0 0" />
      <geometry>
        <mesh
filename="package://cadira_description/meshes/CasterRoda.dae" />
        </geometry>
      </visual>

    <collision>
      <origin xyz="0 0 0" rpy="{M_PI/2} 0 0" />
      <geometry>
        <cylinder length = "0.056" radius = "0.08"/>
      </geometry>
    </collision>
  </link>

  <!-- Definició de l'eslabó de la base Caster -->
  <link name="{caster_prefix}_caster_base">
    <inertial>
      <origin xyz="0 0 0"/>
      <mass value="0.10000000"/>
      <inertia ixx="0.00014957" ixy="-0.00000000"
ixz="0.00005883" iyy="0.00019084" iyz="0.00000000" izz="0.00014979"/>
    </inertial>

    <visual>
      <origin xyz="0 0 0" rpy="0 0 0" />
      <geometry>
        <mesh
filename="package://cadira_description/meshes/CasterBase.dae" />
        </geometry>
      </visual>

    <collision>
      <origin xyz="0 0 0" rpy="0 0 0" />
      <geometry>
        <mesh
filename="package://cadira_description/meshes/CasterBase.dae" />
        </geometry>
      </collision>
    </link>

  <!-- Definició de l'articulació de la roda Caster -->
  <joint name="{caster_prefix}_cwheel" type="continuous">
    <parent link="{caster_prefix}_caster_base"/>
    <child link="{caster_prefix}_caster_link"/>
    <origin xyz="0.06 0 -0.088 " rpy="0 0 0" />
    <axis xyz="0 1 0" rpy="0 0 0" />
  </joint>

```

```

<!-- Definició de l'articulació de la base Caster -->
<joint name="${caster_prefix}_caster" type="continuous">
  <parent link="${parent_link}"/>
  <child link="${caster_prefix}_caster_base"/>
  <xacro:insert_block name="joint_pose"/>
  <axis xyz="0 0 1" rpy="0 0 0" />
  <dynamics friction="0.01" />
</joint>

<!-- Codi específic per a gazebo -->
<gazebo reference="${caster_prefix}_caster_link">
  <mu1 value="${wheel_mu1}"/>
  <mu2 value="${wheel_mu2}"/>
  <kp>${wheel_kp}</kp>
  <kd>${wheel_kd}</kd>
  <dampingFactor>0.05</dampingFactor>
  <fdirl value="-1 0 0"/>
</gazebo>

<gazebo reference="${caster_prefix}_caster_base">
  <mu1 value="${wheel_mu1}"/>
  <mu2 value="${wheel_mu2}"/>
  <kp>${wheel_kp}</kp>
  <kd>${wheel_kd}</kd>
  <fdirl value="1 0 0"/>
</gazebo>

<!--Relacions entre actuadors i articulacions -->
<transmission name="${caster_prefix}_caster_trans"
type="SimpleTransmission">
  <type>transmission_interface/SimpleTransmission</type>
  <actuator name="${caster_prefix}_caster_motor">
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
  <joint name="${caster_prefix}_caster">

<hardwareInterface>hardware_interface/EffortJointInterface</hardwareIn
terface>
  </joint>
</transmission>

  <transmission name="${caster_prefix}_casterbase_trans"
type="SimpleTransmission">
  <type>transmission_interface/SimpleTransmission</type>
  <actuator name="${caster_prefix}_casterbase_motor">
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
  <joint name="${caster_prefix}_cwheel">

<hardwareInterface>hardware_interface/EffortJointInterface</hardwareIn
terface>
  </joint>
</transmission>

</xacro:macro>
</robot>

```

1.2.1.1.3 cadira

En aquest arxiu es defineix la base de la cadira amb les relacions que aquesta té amb les rodes.

```

<?xml version="1.0"?>

<!-- inicialització de variables -->
<robot xmlns:xacro="http://ros.org/wiki/xacro" >

  <xacro:property name="wheel_kp" value="600000" />
  <xacro:property name="wheel_kd" value="3" />
  <xacro:property name="wheel_mu1" value="10000000" />
  <xacro:property name="wheel_mu2" value="10000000" />

  <!--Definició de la macro de la roda Caster -->
  <xacro:macro name="caster" params="caster_prefix parent_link
*joint_pose">
    <!-- Definició de l'eslabó de la roda Caster -->
    <link name="${caster_prefix}_caster_link">
      <inertial>
        <origin xyz="0 0 0"/>
        <mass value="0.10000000"/>
        <inertia ixx="0.00018511" ixy="-0.00000000" ixz="-
0.00000000" iyy="0.00031795" iyz="0.00000000" izz="0.00018511"/>
      </inertial>

      <visual>
        <origin xyz="0 0 0" rpy="0 0 0" />
        <geometry>
          <mesh
filename="package://cadira_description/meshes/CasterRoda.dae" />
          </geometry>
        </visual>

        <collision>
          <origin xyz="0 0 0" rpy="{M_PI/2} 0 0" />
          <geometry>
            <cylinder length = "0.056" radius = "0.08"/>
          </geometry>
        </collision>
      </link>

    <!-- Definició de l'eslabó de la base Caster -->
    <link name="${caster_prefix}_caster_base">
      <inertial>
        <origin xyz="0 0 0"/>
        <mass value="0.10000000"/>
        <inertia ixx="0.00014957" ixy="-0.00000000"
ixz="0.00005883" iyy="0.00019084" iyz="0.00000000" izz="0.00014979"/>
      </inertial>
    </link>
  </macro>

```

```

    <visual>
      <origin xyz="0 0 0" rpy="0 0 0" />
      <geometry>
        <mesh
filename="package://cadira_description/meshes/CasterBase.dae" />
        </geometry>
      </visual>

    <collision>
      <origin xyz="0 0 0" rpy="0 0 0" />
      <geometry>
        <mesh
filename="package://cadira_description/meshes/CasterBase.dae" />
        </geometry>
      </collision>
    </link>

    <!-- Definició de l'articulació de la roda Caster -->
    <joint name="${caster_prefix}_cwheel" type="continuous">
      <parent link="${caster_prefix}_caster_base"/>
      <child link="${caster_prefix}_caster_link"/>
      <origin xyz="0.06 0 -0.088 " rpy="0 0 0" />
      <axis xyz="0 1 0" rpy="0 0 0" />
    </joint>

    <!-- Definició de l'articulació de la base Caster -->
    <joint name="${caster_prefix}_caster" type="continuous">
      <parent link="${parent_link}"/>
      <child link="${caster_prefix}_caster_base"/>
      <xacro:insert_block name="joint_pose"/>
      <axis xyz="0 0 1" rpy="0 0 0" />
      <dynamics friction="0.01" />
    </joint>

    <!-- Codi específic per a gazebo -->
    <gazebo reference="${caster_prefix}_caster_link">
      <mu1 value="${wheel_mu1}"/>
      <mu2 value="${wheel_mu2}"/>
      <kp>${wheel_kp}</kp>
      <kd>${wheel_kd}</kd>
      <dampingFactor>0.05</dampingFactor>
      <fdirl value="-1 0 0"/>
    </gazebo>

    <gazebo reference="${caster_prefix}_caster_base">
      <mu1 value="${wheel_mu1}"/>
      <mu2 value="${wheel_mu2}"/>
      <kp>${wheel_kp}</kp>
      <kd>${wheel_kd}</kd>
      <fdirl value="1 0 0"/>
    </gazebo>

```

```
<!--Relacions entre actuadors i articulacions -->
<transmission name="{caster_prefix}_caster_trans"
type="SimpleTransmission">
  <type>transmission_interface/SimpleTransmission</type>
  <actuator name="{caster_prefix}_caster_motor">
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
  <joint name="{caster_prefix}_caster">
<hardwareInterface>hardware_interface/EffortJointInterface</hardwareIn
terface>
  </joint>
</transmission>

  <transmission name="{caster_prefix}_casterbase_trans"
type="SimpleTransmission">
  <type>transmission_interface/SimpleTransmission</type>
  <actuator name="{caster_prefix}_casterbase_motor">
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
  <joint name="{caster_prefix}_cwheel">
<hardwareInterface>hardware_interface/EffortJointInterface</hardwareIn
terface>
  </joint>
</transmission>

  </xacro:macro>
</robot>
```

1.2.2. Cadira_Control

En aquesta carpeta, s'hi troben els codis usats per a controlar la cadira mitjançant el sensor.

1.2.2.1. Config

En aquesta carpeta s'hi troba el codi usat per a configurar el node que fa la funció de controlador de velocitat de la cadira (en funció de les consignes de velocitat angular i lineal).

1.2.2.1.1 control

```
#S'inicialitza el node que publica les ordres de moviments a les rodes
joint_state_controller:
  type: joint_state_controller/JointStateController
  publish_rate: 50

# S'inicialitza el node que genera el controlador del robot
diferencial
velocity_controller:
  type      : "diff_drive_controller/DiffDriveController"
  left_wheel : 'left_wheel' #Nom de la roda esquerra
  right_wheel : 'right_wheel' #Nom de la roda dreta
  publish_rate: 50           #El missatge es publica amb una
                             #freqüència de 50Hz
  pose_covariance_diagonal : [0.001, 0.001, 1000000.0, 1000000.0,
                             1000000.0, 1000.0]
  twist_covariance_diagonal: [0.001, 0.001, 1000000.0, 1000000.0,
                             1000000.0, 1000.0]

  #Separació entre les rodes motrius i el seu diàmetre.
  wheel_separation : 0.61602
  wheel_radius     : 0.15

  enable_odom_tf : true
  publish_cmd    : true

  # Multiplicadors per a la separació entre rodes i per al radi (per
  # defecte 1)
  wheel_separation_multiplier: 1.0
  wheel_radius_multiplier   : 1.0

  # Timeout de les ordres de velocitat (en segons)
  cmd_vel_timeout: 0.25

  # Nom de la base del robot
  base_frame_id: base_footprint
```

```
# Límits de velocitat i acceleració
linear:
  x:
    has_velocity_limits : true
    max_velocity        : 1.0 # m/s
    min_velocity        : -0.5 # m/s
    has_acceleration_limits: true
    max_acceleration    : 0.2 # m/s^2
    min_acceleration    : -0.2 # m/s^2
    has_jerk_limits     : true
    max_jerk            : 5.0 # m/s^3
  angular:
    z:
      has_velocity_limits : true
      max_velocity        : 3.0 # rad/s
      has_acceleration_limits: true
      max_acceleration    : 0.5 # rad/s^2
      has_jerk_limits     : true
      max_jerk            : 2.5 # rad/s^3
```

1.2.2.2. Scripts

En aquesta carpeta, es troba el codi que té per funció llegir el senyal de control i processar-lo per tal d'obtenir les consignes de velocitat lineal i angular. Aquest codi s'ha programat en python i ha estat dividit per a mantenir-lo endreçat.

1.2.2.2.1 Main/Main_2

Part principal del codi on s'hi calcula la velocitat lineal i angular. Main és el que es fa servir per a les dues cintes. Main 2 es fa servir per al joystick.

```
import time
import sys
import signal
import settings
settings.init()
from datetime import datetime
import threading
import rospy
from geometry_msgs.msg import Twist
from lecturaVoltatges import llegirVoltatges
from filtratge import *
from ContatgeMinMax import *
from menu import *
```

```
#Interrupció de teclat que permet cridar el menú des de dins de una
funció activa
def interrupt_handler(signum, frame):

    if (settings.contantDarrMin):
        settings.aturarThread = True
        settings.indexLectDarrMin = 0

    if (settings.contantDarrMax):
        settings.aturarThread = True
        settings.indexLectDarrMax = 0

    if (settings.contantEsqMin):
        settings.aturarThread = True
        settings.indexLectEsqMin = 0

    if (settings.contantEsqMax):
        settings.aturarThread = True
        settings.indexLectEsqMax = 0

    if (settings.contantDavMin):
        settings.aturarThread = True
        settings.indexLectDavMin = 0

    if (settings.contantDavMax):
        settings.aturarThread = True
        settings.indexLectDavMax = 0

    if (settings.contantDretMin):
        settings.aturarThread = True
        settings.indexLectDretMin = 0

    if (settings.contantDretMax):
        settings.aturarThread = True
        settings.indexLectDretMax = 0

    settings.equilibrarPont = False
    settings.IniciantControl = False
    generarMenu()

#Número de lectures a les que aproximar el voltatge filtrat.
numLectVfilt = 2
#Índex pel filtre de cada element del joystick.
indexVdarrereG = 0
indexVesqG = 0
indexVdavantG = 0
indexVdretG = 0
#Array per emmagatzemar les lectures del joystick.
arrVdarrereG = [0.0 for i in range(numLectVfilt)]
arrVesqG = [0.0 for i in range(numLectVfilt)]
arrVdavantG = [0.0 for i in range(numLectVfilt)]
arrVdretG = [0.0 for i in range(numLectVfilt)]

#Inicialització de la interrupció de teclat
signal.signal(signal.SIGINT, interrupt_handler)
#El primer cop que s'executa el codi s'imprimeix el menú.
genMenu = True

move_cmd = Twist()
```

```
while (True):
    #Si rebem dades del port sèrie, s'executa el codi.
    if (settings.ser.inWaiting()):
        #Processem les dades rebudes pel port serie.
        VxADC = llegirVoltatges(Data =
settings.ser.read_until(b'\r\n\r\n'))

        #Es filtra la dada de cada component del joystick d'acord amb
el nombre de lectures especificades
        darrere = filtrarVoltatge(VxADC[0], arrVdarrereG,
indexVdarrereG, numLectVfilt)
        settings.VgalgaDarrere = darrere.Avg
        arrVdarrereG = darrere.Array
        indexVdarrereG = darrere.Index

        esq = filtrarVoltatge(VxADC[1], arrVesqG, indexVesqG,
numLectVfilt)
        settings.VgalgaEsquerra = esq.Avg
        arrVesqG = esq.Array
        indexVesqG = esq.Index

        davant = filtrarVoltatge(VxADC[2], arrVdavantG, indexVdavantG,
numLectVfilt)
        settings.VgalgaDavant = davant.Avg
        arrVdavantG = davant.Array
        indexVdavantG = davant.Index

        dret = filtrarVoltatge(VxADC[3], arrVdretG, indexVdretG,
numLectVfilt)
        settings.VgalgaDreta = dret.Avg
        arrVdretG = dret.Array
        indexVdretG = dret.Index
        #S'imprimeix el menú en el primer bucle del codi
        if (genMenu):
            generarMenu()
            genMenu = False

        #Si s'habilita la funció equilibrar pont, es mostra el valor
actual llegit per els sensors
        if (settings.equilibrarPont):
            print("V Galga Darrere ", round(settings.VgalgaDarrere,4),
" V Galga Esquerra ", round(settings.VgalgaEsquerra,4), "V Galga
Davant ", round(settings.VgalgaDavant,4), "V Galga Dreta ",
round(settings.VgalgaDreta,4))
```

```

    #Si s'habilita el control, s'executen les següents accions:
    if (settings.IniciantControl):

        #Es calcula el senyal de control en funció del màxim i el
        #minim emmagatzemats entre 0 i 2 (ja que la velocitat màxima és de
        #2m/s).
        VcontrolB = ((settings.VminDarr-
settings.VgalgaDarrere)/(settings.VminDarr-settings.VmaxDarr)) * 2
        VcontrolE = ((settings.VminEsq-
settings.VgalgaEsquerra)/(settings.VminEsq-settings.VmaxEsq)) * 2
        VcontrolF = ((settings.VminDav-
settings.VgalgaDavant)/(settings.VminDav-settings.VmaxDav)) * 2
        VcontrolD = ((settings.VminDret-
settings.VgalgaDreta)/(settings.VminDret-settings.VmaxDret)) * 2

        #S'impedeix que hi hagi valors fora del rang 0-2m/s).
        VclampedB = sorted((0.0, VcontrolB, 2.0))[1]
        VclampedE = sorted((0.0, VcontrolE, 2.0))[1]
        VclampedF = sorted((0.0, VcontrolF, 2.0))[1]
        VclampedD = sorted((0.0, VcontrolD, 2.0))[1]

        #Es calcula la velocitat en funció de la lectura dels
        #sensors de davant i darrere del joystick.
        Vel = (-VclampedF + VclampedB)
        #Es calcula la velocitat angular en funció de la lectura
        #dels sensors esquerra i dret del joystick.
        Ang = (+VclampedD - VclampedE)

        #S'imprimeixen les velocitats en pantalla per a l'usuari.
        print(Vel, " ", Ang)

        #S'inicialitza el node que publica les velocitats.
        pub = rospy.Publisher('/velocity_controller/cmd_vel',
Twist, queue_size=1)
        rospy.init_node('controller')
        rate = rospy.Rate(10) # Es publica les velocitats amb una
#freqüència de 10Hz.
        move_cmd.linear.x = float(Vel)
        move_cmd.angular.z = float(Ang)
        pub.publish(move_cmd) #Es publica les velocitats

        try:
            rate.sleep() #S'espera el temps especificat per la
#freqüència.

        except rospy.ROSInterruptException: #Acció en cas de
#detectar una interrupció de teclat.
            rospy.signal_shutdown("")

```

1.2.2.2.2 Settings

És el codi on s'inicialitzen les variables globals.

```
from menu import *
import serial

#En aquest fragment de codi s'inicialitza les variables globals.

def init():
    global VgalgaDarrere
    global VgalgaEsquerra
    global VgalgaDavant
    global VgalgaDreta

    global numLect
    global aturarThread

    global contantDarrMin
    global arrDarrMin
    global indexLectDarrMin
    global VminDarr

    global contantDarrMax
    global arrDarrMax
    global indexLectDarrMax
    global VmaxDarr

    global contantEsqMin
    global arrEsqMin
    global indexLectEsqMin
    global VminEsq

    global contantEsqMax
    global arrEsqMax
    global indexLectEsqMax
    global VmaxEsq

    global contantDavMin
    global arrDavMin
    global indexLectDavMin
    global VminDav

    global contantDavMax
    global arrDavMax
    global indexLectDavMax
    global VmaxDav

    global contantDretMin
    global arrDretMin
    global indexLectDretMin
    global VminDret

    global contantDretMax
    global arrDretMax
    global indexLectDretMax
    global VmaxDret

    global llista
    global equilibrarPont
    global IniciantControl
    global ser
```

```
VgalgaDarrere = 0.0
VgalgaEsquerra = 0.0
VgalgaDavant = 0.0
VgalgaDreta = 0.0

numLect = 10
aturarThread = False

contantDarrMin = False
arrDarrMin = [0.0 for i in range(numLect)]
indexLectDarrMin = 0
VminDarr = 0.0

contantDarrMax = False
arrDarrMax = [0.0 for i in range(numLect)]
indexLectDarrMax = 0
VmaxDarr = 0.0

contantEsqMin = False
arrEsqMin = [0.0 for i in range(numLect)]
indexLectEsqMin = 0
VminEsq = 0.0

contantEsqMax = False
arrEsqMax = [0.0 for i in range(numLect)]
indexLectEsqMax = 0
VmaxEsq = 0.0

contantDavMin = False
arrDavMin = [0.0 for i in range(numLect)]
indexLectDavMin = 0
VminDav = 0.0

contantDavMax = False
arrDavMax = [0.0 for i in range(numLect)]
indexLectDavMax = 0
VmaxDav = 0.0

contantDretMin = False
arrDretMin = [0.0 for i in range(numLect)]
indexLectDretMin = 0
VminDret = 0.0

contantDretMax = False
arrDretMax = [0.0 for i in range(numLect)]
indexLectDretMax = 0
VmaxDret = 0.0
```

```

llista = {
    'E': ('Equilibrar Pont', EqPont),
    '#Mb': ('Actualitzar Màxim', obtenirMaxDarr),
    '#mb': ('Actualitzar Mínim', obtenirMinDarrere),
    'Me': ('Actualitzar Màxim', obtenirMaxEsq),
    'me': ('Actualitzar Mínim', obtenirMinEsq),
    '#Mf': ('Actualitzar Màxim', obtenirMaxDav),
    '#mf': ('Actualitzar Mínim', obtenirMinDavant),
    'Md': ('Actualitzar Màxim', obtenirMaxDret),
    'md': ('Actualitzar Mínim', obtenirMinDret),
    'A': ('Iniciar el control', ControlantCadira),
    'Q': ('Sortir', Quit)
}

equilibrarPont = False
IniciantControl = False

#Inicialització del port sèrie
Port_ADC = "/dev/ttyUSB0" #input('Port serie per el ADC?\n')
ser = serial.Serial(Port_ADC, 115200)

```

1.2.2.3 Menú

S'hi troba el codi necessari per a crear un menú funcional.

```

import settings
from ContatgeMinMax import*
import sys

#Funció que imprimeix el menú i espera una acció.
def generarMenu():
    seleccio = None
    mostrarMenu()
    seleccio = llegirSeleccio()
    executarAccio(seleccio)

#Funció per a visualitzar el menú.
def mostrarMenu():
    print('Menu: ')
    for i in sorted(settings.llista):
        print(f' {i}) {settings.llista[i][0]}')

#Funció per llegir l'element seleccionat del menú
def llegirSeleccio():
    a = None
    #Si no es selecciona cap vàlid tornem a imprimir el menú.
    while (a) not in settings.llista:
        a = input('Escollir Acció ')
    return a

#Funció que executa l'acció escollida pel menú
def executarAccio(seleccio):
    settings.llista[seleccio][1]()

```

```
#Si es selecciona l'acció equilibrar el pont, s'habilita el equilibrat
mitjançant la variable corresponent en aquesta funció
def EqPont():
    settings.equilibrarPont = True
    settings.ser.flushInput()

#Si es selecciona l'acció Activar control, s'habilita el control
mitjançant la variable corresponent en aquesta funció
def ControlantCadira():
    settings.IniciantControl = True
    settings.ser.flushInput()

#Si es selecciona sortir, es tanca tots els processos del codi.
#Funcio per sortir
def Quit():
    sys.exit(0)
```

1.2.2.2.4 LecturaVoltatges

En aquest fragment de codi, s'hi processa el voltatge llegit de l'ADC.

```
import re

#Es separa les dades de tipus string rebudes en el port sèrie i es
converteixen en tipus float
def DemuxData(RawData):

    V0 = 0.0
    V1 = 0.0
    V2 = 0.0
    V3 = 0.0
    V4 = 0.0
    V5 = 0.0
    V6 = 0.0
    V7 = 0.0
    V8 = 0.0
    V9 = 0.0

    DataLines = RawData.splitlines()

    for x in range(len(DataLines)):

        DataString = DataLines[x].decode("utf-8")
        DataVect = re.split(':', DataString)
        Name = DataVect[0]

        if Name == "CH0":
            ADC0 = DataVect[1]
            V0st = DataVect[2]
            V0st = V0st.replace("v", "")
            V0 = float(V0st)

        if Name == "CH1":
            ADC1 = DataVect[1]
            V1st = DataVect[2]
            V1st = V1st.replace("v", "")
            V1 = float(V1st)
```

```
if Name == "CH2":
    ADC2 = DataVect[1]
    V2st = DataVect[2]
    V2st = V2st.replace("v", "")
    V2 = float(V2st)

if Name == "CH3":
    ADC3 = DataVect[1]
    V3st = DataVect[2]
    V3st = V3st.replace("v", "")
    V3 = float(V3st)

if Name == "CH4":
    ADC4 = DataVect[1]
    V4st = DataVect[2]
    V4st = V4st.replace("v", "")
    V4 = float(V4st)

if Name == "CH5":
    ADC5 = DataVect[1]
    V5st = DataVect[2]
    V5st = V5st.replace("v", "")
    V5 = float(V5st)

if Name == "CH6":
    ADC6 = DataVect[1]
    V6st = DataVect[2]
    V6st = V6st.replace("v", "")
    V6 = float(V6st)

if Name == "CH7":
    ADC7 = DataVect[1]
    V7st = DataVect[2]
    V7st = V7st.replace("v", "")
    V7 = float(V7st)

if Name == "CH8":
    ADC8 = DataVect[1]
    V8st = DataVect[2]
    V8st = V8st.replace("v", "")
    V8 = float(V8st)

if Name == "CH9":
    ADC9 = DataVect[1]
    V9st = DataVect[2]
    V9st = V9st.replace("v", "")
    V9 = float(V9st)

return V0, V1, V2, V3, V4, V5, V6, V7, V8, V9
```

```
#Es calcula el voltatge de cada sensor com la resta de la lectura de
dues entrades del adc (per a permetre llegir valors negatius)
def llegirVoltatges(Data):
```

```
    VGalgaDarrere = None
    VGalgaEsquerra = None
    VGalgaDavant = None
    VGalgaDret = None
```

```
    if (Data):
        Vx = DemuxData(Data)
        VGalgaDarrere = Vx[1] - Vx[0]
        VGalgaEsquerra = Vx[3] - Vx[2]
        VGalgaDavant = Vx[5] - Vx[4]
        VGalgaDret = Vx[7] - Vx[6]
    return VGalgaDarrere, VGalgaEsquerra, VGalgaDavant, VGalgaDret
```

1.2.2.2.5 Filtratge

Aquesta funció serveix per aplicar el filtre.

```
#Definició de la classe Filtratge
```

```
class Filtratge:
```

```
    def __init__(self, Array, Index, Avg):
        self.Array = Array
        self.Index = Index
        self.Avg = Avg
```

```
#Funció per filtrar lectures en base a el número de valors a utilitzar
per al filtre
```

```
def filtrarVoltatge(ValorLlegit, Array, Index, numLectures):
```

```
    Array[Index] = ValorLlegit
    Index = Index + 1
    if (Index >= numLectures):
        Index = 0
    Total = 0.0
    for i in Array:
        Total = Total + i
    Avg = Total/numLectures
```

```
ValorFiltrat = Filtratge(Array, Index, Avg)
return ValorFiltrat
```

1.2.2.2.6 ContatgeMinMax

En aquestes funcions s'hi emmagatzema els valors dels voltatges màxims i mínims dels sensors.

```
import threading
from filtratge import *
import settings

#Funcions per obtenir i filtrar els maxims i minims de cada sensor.
def obtenirMinDarrere():

    settings.contantDarrMin = True

    t1 = threading.Timer(1.0, obtenirMinDarrere)

    if (not settings.aturarThread):

        minDat = filtrarVoltatge(settings.VgalgaDarrere,
settings.arrDarrMin, settings.indexLectDarrMin, settings.numLect)
        settings.VminDarr = minDat.Avg
        settings.arrDarrMin = minDat.Array
        settings.indexLectDarrMin = minDat.Index

        print(settings.VminDarr, ";", settings.VgalgaDarrere, ";",
settings.indexLectDarrMin)

    if ((settings.indexLectDarrMin > 0) and not
settings.aturarThread):
        t1.start()
    if (settings.aturarThread or (settings.indexLectDarrMin==0)):
        settings.aturarThread = False
        settings.contantDarrMin = False
```

```
def obtenirMaxDarr():

    settings.contantDarrMax = True

    t1 = threading.Timer(1.0, obtenirMaxDarr)

    if (not settings.aturarThread):

        minDat = filtrarVoltatge(settings.VgalgaDarrere,
settings.arrDarrMax, settings.indexLectDarrMax, settings.numLect)
        settings.VmaxDarr = minDat.Avg
        settings.arrDarrMax = minDat.Array
        settings.indexLectDarrMax = minDat.Index

        print(settings.VmaxDarr, ";", settings.VgalgaDarrere, ";",
settings.indexLectDarrMax)

    if ((settings.indexLectDarrMax > 0) and not
settings.aturarThread):
        t1.start()
    if (settings.aturarThread or (settings.indexLectDarrMax==0)):
        settings.aturarThread = False
        settings.contantDarrMax = False

def obtenirMinEsq():

    settings.contantEsqMin = True

    t1 = threading.Timer(1.0, obtenirMinEsq)

    if (not settings.aturarThread):

        minDat = filtrarVoltatge(settings.VgalgaEsquerra,
settings.arrEsqMin, settings.indexLectEsqMin, settings.numLect)
        settings.VminEsq = minDat.Avg
        settings.arrEsqMin = minDat.Array
        settings.indexLectEsqMin = minDat.Index

        print(settings.VminEsq, ";", settings.VgalgaEsquerra, ";",
settings.indexLectEsqMin)

    if ((settings.indexLectEsqMin > 0) and not settings.aturarThread):
        t1.start()
    if (settings.aturarThread or (settings.indexLectEsqMin==0)):
        settings.aturarThread = False
        settings.contantEsqMin = False
```

```
def obtenirMinDavant():

    settings.contantDavMin = True

    t1 = threading.Timer(1.0, obtenirMinDavant)

    if (not settings.aturarThread):

        minDat = filtrarVoltatge(settings.VgalgaDavant,
settings.arrDavMin, settings.indexLectDavMin, settings.numLect)
        settings.VminDav = minDat.Avg
        settings.arrDavMin = minDat.Array
        settings.indexLectDavMin = minDat.Index

        print(settings.VminDav, ";", settings.VgalgaDavant, ";",
settings.indexLectDavMin)

    if ((settings.indexLectDavMin > 0) and not settings.aturarThread):
        t1.start()
    if (settings.aturarThread or (settings.indexLectDavMin==0)):
        settings.aturarThread = False
        settings.contantDavMin = False

def obtenirMaxDav():

    settings.contantDavMax = True

    t1 = threading.Timer(1.0, obtenirMaxDav)

    if (not settings.aturarThread):

        minDat = filtrarVoltatge(settings.VgalgaDavant,
settings.arrDavMax, settings.indexLectDavMax, settings.numLect)
        settings.VmaxDav = minDat.Avg
        settings.arrDavMax = minDat.Array
        settings.indexLectDavMax = minDat.Index

        print(settings.VmaxDav, ";", settings.VgalgaDavant, ";",
settings.indexLectDavMax)

    if ((settings.indexLectDavMax > 0) and not settings.aturarThread):
        t1.start()
    if (settings.aturarThread or (settings.indexLectDavMax==0)):
        settings.aturarThread = False
        settings.contantDavMax = False
```

```
def obtenirMinDret():

    settings.contantDretMin = True

    t1 = threading.Timer(1.0, obtenirMinDret)

    if (not settings.aturarThread):

        minDat = filtrarVoltatge(settings.VgalgaDreta,
settings.arrDretMin, settings.indexLectDretMin, settings.numLect)
        settings.VminDret = minDat.Avg
        settings.arrDretMin = minDat.Array
        settings.indexLectDretMin = minDat.Index

        print(settings.VminDret, ";", settings.VgalgaDreta, ";",
settings.indexLectDretMin)

    if ((settings.indexLectDretMin > 0) and not
settings.aturarThread):
        t1.start()
    if (settings.aturarThread or (settings.indexLectDretMin==0)):
        settings.aturarThread = False
        settings.contantDretMin = False

def obtenirMaxDret():

    settings.contantDretMax = True

    t1 = threading.Timer(1.0, obtenirMaxDret)

    if (not settings.aturarThread):

        minDat = filtrarVoltatge(settings.VgalgaDreta,
settings.arrDretMax, settings.indexLectDretMax, settings.numLect)
        settings.VmaxDret = minDat.Avg
        settings.arrDretMax = minDat.Array
        settings.indexLectDretMax = minDat.Index

        print(settings.VmaxDret, ";", settings.VgalgaDreta, ";",
settings.indexLectDretMax)

    if ((settings.indexLectDretMax > 0) and not
settings.aturarThread):
        t1.start()
    if (settings.aturarThread or (settings.indexLectDretMax==0)):
        settings.aturarThread = False
        settings.contantDretMax = False
```

1.2.3. Cadira_viz

En aquest directori, s'hi troba l'arxiu que es fa servir per executar la simulació (dintre de la carpeta launch).

1.2.3.1. Launch

Carpeta emprada en ROS usada per a contenir tots els fitxers relacionats amb la execució dels models.

1.2.3.1.1 Cadira_vis

Aquest codi té format XML i en ell s'inicia ROS i es llança el model de la cadira.

```
<?xml version="1.0" encoding="UTF-8"?>
<launch>
  <arg name="paused" default="false"/>
  <arg name="use_sim_time" default="true"/>
  <arg name="robot_namespace" default="/" />

  <!-- S'inicia un món buit en gazebo -->
  <include file="$(find gazebo_ros)/launch/empty_world.launch">
    <arg name="paused" value="$(arg paused)"/>
    <arg name="use_sim_time" value="$(arg use_sim_time)"/>
    <arg name="gui" value="true"/>
    <arg name="headless" value="false"/>
    <arg name="debug" value="false"/>
    <arg name="verbose" default="false"/>
    <arg name="gui_required" value="true"/>
  </include>

  <!-- S'inclou el model de la cadira en el món buit i s'inicia el
  controlador de velocitat -->
  <group ns="$(arg robot_namespace)">
    <param name="robot_description" command="$(find xacro)/xacro
'$(find cadira_description)/urdf/cadira.urdf.xacro' --inorder
robot_namespace:=$(arg robot_namespace)"/>
    <node name="cadira_spawn" pkg="gazebo_ros" type="spawn_model"
output="screen" args="  -z 0.0
                        -urdf -param robot_description
                        -model $(arg robot_namespace)"/>

    <rosparam command="load" file="$(find
cadira_control)/config/control.yaml"/>

    <node name="base_controller_spawner" pkg="controller_manager"
type="spawner"
          args="  joint_state_controller
                velocity_controller
                "/>

    <node name="robot_state_publisher" pkg="robot_state_publisher"
type="robot_state_publisher">
    </node>
  </group>
</launch>
```