

TAMPICO, TAMAULIPAS A **08 DE MARZO DE 2023**



UAT
Universidad Autónoma
de Tamaulipas



Facultad de Ingeniería
Tampico

PORTAFOLIO DE EVIDENCIAS

“UNIDAD 2”

DISEÑO ELECTRONICO BASADO EN SISTEMAS EMBEBIDOS

Profesor: Dr. García Ruiz Alejandro Humberto

8vo Semestre – Grupo “I”

2023-1

Autor de Entregas Individuales:

Ruiz García Emmanuel Alejandro – 2193330288

Autores de Entregas en Equipo:

Hernández Santos Reyna Margarita – 2193330264

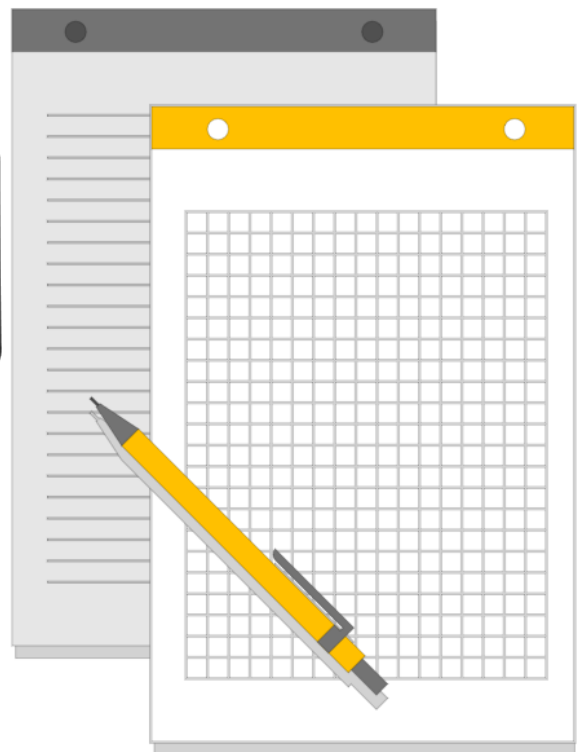
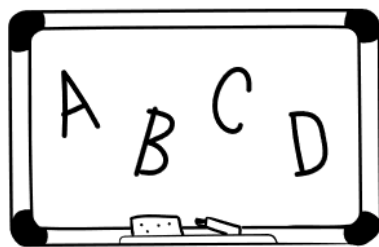
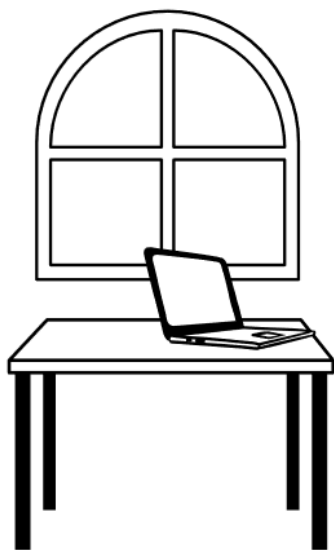
Salas Alardin Luis Fernando – 2111340059

Luna Sánchez Juan Pablo – 2163216015

ÍNDICE

1. ACTIVIDADES DE CLASE	3
1.1 ENTREGAS EN EQUIPO	4
1.2 ENTREGAS INDIVIDUALES.....	33
2. TAREAS E INVESTIGACIONES	34
2.1 ENTREGAS EN EQUIPO	35
2.2 ENTREGAS INDIVIDUALES.....	49
3. PROGRAMAS.....	51
3.1 ENTREGAS EN EQUIPO	52
3.2 ENTREGAS INDIVIDUALES.....	53
4. PRÁCTICAS	54
4.1 ENTREGAS EN EQUIPO	55
4.1.1 <i>Desarrollo de las Prácticas</i>	55
4.2 ENTREGAS INDIVIDUALES.....	62
5. PROYECTO.....	63

ACTIVIDADES



ACTIVIDADES DE CLASE

1.1 Entregas en Equipo

EJERCICIO 1.

```
P_1_SendToPython.py x
1
2 import serial as s #pyserial
3
4 arduino = None
5
6 arduino = s.Serial("COM5", baudrate=9600, timeout=1)
7
8
9 while True:
10     cadena = arduino.readline()
11     #print(cadena) ## imprime como... b'419\r\n'
12     cadena = cadena.decode()
13     #print(cadena) ## imprime como..419 con saltos de linea entre impresion
14     cadena = cadena.replace("\n", "")
15     cadena = cadena.replace("\r", "")
16     print(cadena * 3) ##<----
17
```

EJERCICIO 2.

```
P_2_SendDataPotentiometer.py x
1
2 import serial as s
3
4 arduino = None
5
6 arduino = s.Serial("COM5", baudrate=9600, timeout=1)
7
8 lista = []
9 tolecturas = 5
10 i = 0
11 while i < tolecturas:
12     cadena = arduino.readline()
13     #print(cadena) ## imprime como... b'419\r\n'
14     cadena = cadena.decode()
15     #print(cadena) ## imprime como..419 con saltos de linea entre impresion
16     cadena = cadena.replace("\n", "")
17     cadena = cadena.replace("\r", "")
18     #print(cadena) ##<----
19     lista.append(cadena)
20     i+=1
21
22 print(lista)
23
24 lista = list(map(int, lista))
25
26 print(lista)
```

EJERCICIO 3.

```

1  import serial as s
2
3  arduino = None
4
5  arduino = s.Serial("COM6", baudrate=9600, timeout=1)
6
7  lista = []
8  totlecturas = 5
9  i = 0
10 while i < totlecturas:
11     cadena = arduino.readline()
12     #print(cadena) ## imprime como.... b'419\r\n'
13     cadena = cadena.decode()
14     #print(cadena) ## imprime como...419 con saltos de linea entre impresion
15     cadena = cadena.replace("\n", "")
16     cadena = cadena.replace("\r", "")
17     #print(cadena) ##<<----
18     if cadena!="":
19         lista.append(cadena)
20     i+=1
21
22 lista = list(map(int, lista))
23 print(lista)
24
25 #archivo = open("C:\Users\Reyna\Documents\SE_I_U2_EQ_3\Archivos\P_3_SendDP_Promedio.csv", "w")
26 archivo = open("../Archivos/P_3_SendDP_Promedio.csv", "w")
27
28 for lectura in lista:#
29     #print(lectura)
30     archivo.write(str(lectura) + ",")
31
32

```

EJERCICIO 4.

```

P_4_SendDP_ValMenor.py x
1  import serial as s
2
3  arduino = None
4
5  arduino = s.Serial("COM10", baudrate=9600, timeout=1)
6
7  lista = []
8  totlecturas = 30
9  i = 0
10 while i < totlecturas:
11     cadena = arduino.readline()
12     #print(cadena) ## imprime como... b'419\r\n'
13     cadena = cadena.decode()
14     #print(cadena) ## imprime como... 419 con saltos de linea entre impresion
15     cadena = cadena.replace("\n", " ")
16     cadena = cadena.replace("\r", " ")
17     #print(cadena) ##<----
18     if cadena!="":
19         lista.append(cadena)
20         i+=1
21
22 lista = list(map(int, lista))
23 print(lista)
24
25 archivo = open("../Archivos/P_4_SendDP_ValMenor.csv", "w")
26
27 for lectura in lista:#
28     #print(lectura)
29     archivo.write(str(lectura) + ",")
30
31
32 archivo.flush()

```

EJERCICIO 5.

```

P_5_SendDP_ValMayor.py x
1  import serial as s
2
3  arduino = None
4
5  arduino = s.Serial("COM10", baudrate=9600, timeout=1)
6
7  lista = []
8  totlecturas = 0
9  i = 30
10 while i > totlecturas:
11     cadena = arduino.readline()
12     #print(cadena) ## imprime como... b'419\r\n'
13     cadena = cadena.decode()
14     #print(cadena) ## imprime como... 419 con saltos de linea entre impresion
15     cadena = cadena.replace("\n", "")
16     cadena = cadena.replace("\r", "")
17     #print(cadena) ##<<---
18     if cadena!="":
19         lista.append(cadena)
20         i-=1
21     #totlecturas -=1
22 lista = list(map(int, lista))
23 print(lista)
24
25 archivo = open("../Archivos/P_5_SendDP_ValMayor.csv", "w")
26
27 for lectura in lista:#
28     #print(lectura)
29     archivo.write(str(lectura) + ",")
30
31
32     archivo.flush()

```

EJERCICIO 6.

```
P_6_GUI_Python_Arduino.py x
1  import sys
2
3      import serial as conectaX #para trabajar con Arduino
4
5  from PyQt5 import uic, QtWidgets, QtCore
6
7      qtCreatorFile = "P_6_GUI_Python_Arduino.ui" # Nombre del archivo aqui.
8
9      Ui_MainWindow, QtBaseClass = uic.loadUiType(qtCreatorFile)
10
11      Emmanuel Alejandro Ruiz Garcia
12  class MyApp(QtWidgets.QMainWindow, Ui_MainWindow):
13      Emmanuel Alejandro Ruiz Garcia
14      def __init__(self):
15          QtWidgets.QMainWindow.__init__(self)
16          Ui_MainWindow.__init__(self)
17          self.setupUi(self)
18
19          self.txt_puerto.setText("")
20
21          # Área de los Signals
22          self.arduino = None
23          self.btn_accion.clicked.connect(self.accion)
24          self.segundoPlano = QtCore.QTimer()
25          self.segundoPlano.timeout.connect(self.control)
26
27
```



```

31 def control(self):
32     try:
33         valor = self.arduino.readline().decode()
34         valor = valor.replace("\r", "")
35         valor = valor.replace("\n", "")
36         print(valor)
37     except Exception as error:
38         print(error)
39
40 def accion(self):
41     try:
42         if self.arduino == None:
43             puerto = self.txt_puerto.text()
44             self.arduino = conectaX.Serial(puerto, baudrate=9600, timeout=1)
45             self.btn_accion.setText("DESCONECTAR")
46             self.btn_accion.setText("Conectado")
47             self.segundoPlano.start(100)
48         elif self.arduino.isOpen():
49             self.segundoPlano.stop()
50             self.arduino.close()
51             self.btn_accion.setText("CONECTAR")
52             self.txt_estado.setText("DESCONECTADO")
53         else:
54             self.arduino.open()
55             self.btn_accion.setText("CONECTAR")
56             self.btn_accion.setText("DESCONECTAR")
57             self.segundoPlano.start(100)
58     except Exception as error:
59         print(error)
60
61 if __name__ == "__main__":
62     app = QtWidgets.QApplication(sys.argv)
63     window = MyApp()
64     window.show()
65     sys.exit(app.exec_())

```

EJERCICIO 7.

```

1  import sys
2
3  import serial as conectaX #para trabajar con Arduino
4
5  from PyQt5 import uic, QtWidgets, QtCore
6
7  qtcCreatorFile = "P_6_GUI_Python_Arduino.ui" # Nombre del archivo aqui.
8
9  Ui_MainWindow, QtBaseClass = uic.loadUiType(qtcCreatorFile)
10
11
12  class MyApp(QtWidgets.QMainWindow, Ui_MainWindow):
13      def __init__(self):
14          QtWidgets.QMainWindow.__init__(self)
15          Ui_MainWindow.__init__(self)
16          self.setupUi(self)
17
18          #self.txt_puerto.setText("")
19
20          # Área de los Signals
21          self.arduino = None
22          self.btn_accion.clicked.connect(self.accion)
23          self.segundoPlano = QtCore.QTimer()
24          self.segundoPlano.timeout.connect(self.control)
25
26
27
28
29      # Área de los Slots
30      def control_led(self):
31          if not self.arduino == None:
32              if self.arduino.isOpen():
33                  texti = self.btn_control_led.text()
34                  if texti == "APRENDER":
35                      self.btn_control_led.setText("APAGAR")
36                      action = "1".encode()
37                      self.arduino.write(action)
38                  else:
39                      self.btn_control_led.setText("PRENDER")
40                      self.arduino.write("0").encode()
41
42      def control(self):
43          try:
44              valor = self.arduino.readline().decode()
45              valor = valor.replace("\n", "")
46              valor = valor.replace("\n", "")
47              print(valor)
48          except Exception as error:
49              print(error)
50
51      def accion(self):
52          try:
53              if self.arduino == None:
54                  puerto = self.txt_puerto.text()
55                  self.arduino = conectaX.Serial(puerto, baudrate=9600, timeout=1)
56                  self.btn_accion.setText("DESCONECTAR")
57                  self.btn_accion.setText("Conectado")
58                  # self.segundoPlano.start(100)
59              elif self.arduino.isOpen():

```

```

49 Emmanuel Alejandro Ruiz Garcia
50 def accion(self):
51     try:
52         if self.arduino == None:
53             puerto = self.txt_puerto.text()
54             self.arduino = conectaX.Serial(puerto, baudrate = 9600, timeout = 1)
55             self.btn_accion.setText("DESCONECTAR")
56             self.btn_accion.setText("Conectado")
57             # self.segundoPlano.start(100)
58         elif self.arduino.isOpen():
59             #self.segundoPlano.stop()
60             self.arduino.close()
61             self.btn_accion.setText("CONECTAR")
62             self.txt_estado.setText("DESCONECTADO")
63         else:
64             self.arduino.open()
65             self.btn_accion.setText("CONECTAR")
66             self.btn_accion.setText("DESCONECTAR")
67             #SSself.segundoPlano.start(100)
68     except Exception as error:
69         print(error)
70
71
72 if __name__ == "__main__":
73     app = QtWidgets.QApplication(sys.argv)
74     window = MyApp()
75     window.show()
76     sys.exit(app.exec_())
77

```

EJERCICIO 8.

```

1  import sys
2
3  import serial as conectaX #para trabajar con Arduino
4
5  from PyQt5 import uic, QtWidgets, QtCore
6
7  qtCreatorFile = "P_6_GUI_Python_Arduino.ui" # Nombre del archivo aquí.
8
9  Ui_MainWindow, QtBaseClass = uic.loadUiType(qtCreatorFile)
10
11
12  class MyApp(QtWidgets.QMainWindow, Ui_MainWindow):
13      def __init__(self):
14          QtWidgets.QMainWindow.__init__(self)
15          Ui_MainWindow.__init__(self)
16          self.setupUi(self)
17
18          #self.txt_puerto.setText("")
19
20          # Área de los Signals
21          self.arduino = None
22          self.btn_accion.clicked.connect(self.accion)
23          self.segundoPlano = QtCore.QTimer()
24          self.segundoPlano.timeout.connect(self.control)
25
26
27
28
29  # Área de los Slots
30
31  def control_led(self):
32      if not self.arduino == None:
33          if self.arduino.isOpen():
34              texto = self.btn_control_led.text()
35              if texto == "APRENDER":
36                  self.btn_control_led.setText("APAGAR")
37                  action = "1".encode()
38                  self.arduino.write(action)
39              else:
40                  self.btn_control_led.setText("PRENDER")
41                  self.arduino.write("0").encode()
42
43  def control(self):
44      try:
45          # estamos leyendo pero puede que no haya nada.
46          if self.arduino.inWaiting(): #Señal.avaible() > 0
47              valor = self.arduino.readline().decode() #arduino le esta mandando informacion
48              valor = valor.replace("\r", "")
49              valor = valor.replace("\n", "")
50              if valor != "":
51                  print(valor)
52                  self.lw_datos.addItem(valor) #str
53                  #currentRow ...
54                  self.lw_datos.setCurrentRow(self.lw_datos.count()-1) #str
55                  print(valor)#imprime a consola
56                  #self.lw_datos.addItem(valor)
57                  #self.lw_datos.setCurrentRow(self.lw_datos.count()-1)
58      except Exception as error:
59          print(error)

```

```

Emmanuel Alejandro Ruiz Garcia
60 def accion(self):
61     try:
62         if self.arduino == None:
63             puerto = self.txt_puerto.text()
64             self.arduino = conectaX.Serial(puerto, baudrate = 9600, timeout = 1)
65             self.btn_accion.setText("DESCONECTAR")
66             self.btn_accion.setText("Conectado")
67             self.segundoPlano.start(100)
68         elif self.arduino.isOpen():
69             self.segundoPlano.stop()
70             self.arduino.close()
71             self.btn_accion.setText("CONECTAR")
72             self.txt_estado.setText("DESCONECTADO")
73         else:
74             self.arduino.open()
75             self.btn_accion.setText("CONECTAR")
76             self.btn_accion.setText("DESCONECTAR")
77             #SSself.segundoPlano.start(100)
78     except Exception as error:
79         print(error)
80
81
82
83 if __name__ == "__main__":
84     app = QtWidgets.QApplication(sys.argv)
85     window = MyApp()
86     window.show()
87     sys.exit(app.exec_())

```

EJERCICIO 9.

```
P_9_GUI_ControlLED_SolicitarConfirmacion.py ×
1  import sys
2  import serial as conectaX
3
4  from PyQt5 import uic, QtWidgets, QtCore
5
6  qtCreatorFile = "P_8_GUI_ControlLED_confirmacion.ui" # Nombre del archivo aquí.
7
8  Ui_MainWindow, QtBaseClass = uic.loadUiType(qtCreatorFile)
9
10
11  class MyApp(QtWidgets.QMainWindow, Ui_MainWindow):
12      def __init__(self):
13          QtWidgets.QMainWindow.__init__(self)
14          Ui_MainWindow.__init__(self)
15          self.setupUi(self)
16
17          # Área de los Signals
18          self.txt_puerto.setText("COM6")
19          self.arduino = None
20          self.btn_accion.clicked.connect(self.accion)
21
22          self.btn_control_led.clicked.connect(self.control_led)
23
24          self.segundoPlane = QtCore.QTimer()
25          self.segundoPlane.timeout.connect(self.control)
26
27          # Área de los Slots
28          def control_led(self):
29              if not self.arduino == None:
```

```

26
27 # Área de los Slots
   ReynaSantos501
28 def control_led(self):
29     if not self.arduino == None:
30         if self.arduino.isOpen():
31             texto = self.btn_control_led.text()
32             if texto == "PRENDER":
33                 self.btn_control_led.setText("APAGAR")
34                 action = "1".encode()
35                 self.arduino.write(action)
36             else:
37                 self.btn_control_led.setText("PRENDER")
38                 self.arduino.write("0".encode())
39
40
   ReynaSantos501
41 def control(self):
42     try:
43         if self.arduino.inWaiting(): #Serial.available()>0
44             valor = self.arduino.readline().decode()
45             valor = valor.replace("\r", "")
46             valor = valor.replace("\n", "")
47             if valor != "": #Solo agregamos cadenas que no esten vacias
48                 print(valor) #consola
49                 self.lw_datos.addItem(valor) #string
50                 self.lw_datos.setCurrentRow(self.lw_datos.count()-1)
51     except Exception as error:
52         print(error)
53
   ReynaSantos501
54 def accion(self):
55     try:
56         if self.arduino == None:
57             puerto = self.txt_puerto.text()
58             self.arduino = conectaX.Serial(puerto, baudrate=9600, timeout=1)
59             self.btn_accion.setText("DESCONECTAR")
60             self.txt_estado.setText("CONECTADO")
61             self.segundoPlane.start(10)
62         elif self.arduino.isOpen():
63             self.segundoPlane.stop()
64             self.arduino.close()
65             self.btn_accion.setText("CONECTAR")
66             self.txt_estado.setText("DESCONECTADO")
67         else:
68             self.arduino.open()
69             self.btn_accion.setText("DESCONECTAR")
70             self.txt_estado.setText("CONECTADO")
71             self.segundoPlane.start(10)
72     except Exception as error:
73         print(error)
74
75 if __name__ == "__main__":
76     app = QtWidgets.QApplication(sys.argv)
77     window = MyApp()
78     window.show()
79     sys.exit(app.exec_())

```

EJERCICIO 10.

```
P_10_SerializacionSensores.py x
1  import sys
2
3  import serial as conectaX #para trabajar con Arduino
4
5  from PyQt5 import uic, QtWidgets, QtCore
6
7  qtCreatorFile = "P_6_GUI_Python_Arduino.ui" # Nombre del archivo aqui.
8
9  Ui_MainWindow, QtBaseClass = uic.loadUiType(qtCreatorFile)
10
11
12  class MyApp(QtWidgets.QMainWindow, Ui_MainWindow):
13      def __init__(self):
14          QtWidgets.QMainWindow.__init__(self)
15          Ui_MainWindow.__init__(self)
16          self.setupUi(self)
17
18          #self.txt_puerto.setText("")
19
20          # Área de los Signals
21          self.arduino = None
22          self.btn_accion.clicked.connect(self.accion)
23          self.segundoPlano = QtCore.QTimer()
24          self.segundoPlano.timeout.connect(self.control)
25
26
27
28
29  # Área de los Slots
30
```



```

29 # Área de los Slots
30 Emmanuel Alejandro Ruiz Garcia
31 def control_led(self):
32     if not self.arduino == None:
33         if self.arduino.isOpen():
34             texto = self.btn_control_led.text()
35             if texto == "APRENDER":
36                 self.btn_control_led.setText("APAGAR")
37                 action = "1".encode()
38                 self.arduino.write(action)
39             else:
40                 self.btn_control_led.setText("PRENDER")
41                 self.arduino.write("0".encode())
42 Emmanuel Alejandro Ruiz Garcia
43 def control(self):
44     try:
45         # estamos leyendo pero puede que no haya nada.
46         if self.arduino.inWaiting(): #Serial.available() > 0
47             valor = self.arduino.readline().decode() #arduino le esta mandando informacion , pero no a la misma velocidad que py
48             valor = valor.replace("\n", "")
49             valor = valor.replace("\r", "")
50             if valor != "":
51                 print(valor)
52                 self.lw_datos.addItem(valor) #str
53                 #currentRow ...
54                 self.lw_datos.setCurrentRow(self.lw_datos.count()-1) #str
55                 print(valor) #imprime a consola
56
57                 if valor[0] == "E" and valor[-1] == "J":
58                     pass
59
60                 #self.lw_datos.addItem(valor)
61                 #self.lw_datos.setCurrentRow(self.lw_datos.count()-1)
62 except Exception as error:
63     print(error)
64
65 Emmanuel Alejandro Ruiz Garcia
66 def accion(self):
67     try:
68         if self.arduino == None:
69             puerto = self.txt_puerto.text()
70             self.arduino = conectaX.Serial(puerto, baudrate=9600, timeout=1)
71             self.btn_accion.setText("DESCONECTAR")
72             self.btn_accion.setText("Conectado")
73             self.segundoPlano.start(100)
74         elif self.arduino.isOpen():
75             self.segundoPlano.stop()
76             self.arduino.close()
77             self.btn_accion.setText("CONECTAR")
78             self.txt_estado.setText("DESCONECTADO")
79         else:
80             self.arduino.open()
81             self.btn_accion.setText("CONECTAR")
82             self.btn_accion.setText("DESCONECTAR")
83             self.segundoPlano.start(100)
84     except Exception as error:
85         print(error)

```

EJERCICIO 11.

```

P_11_Serializacion.py x
GE_1_U2_EQ_3\Python rt sys
2 import serial as conectaX
3
4 from PyQt5 import uic, QtWidgets, QtCore
5
6 qtcCreatorFile = "P_10hastaXX_SerializacioSensores.ui" # Nombre del archivo aqui.
7
8 Ui_MainWindow, QtBaseClass = uic.loadUiType(qtcCreatorFile)
9
10
11 class MyApp(QtWidgets.QMainWindow, Ui_MainWindow):
12     def __init__(self):
13         QtWidgets.QMainWindow.__init__(self)
14         Ui_MainWindow.__init__(self)
15         self.setupUi(self)
16
17         # Área de los Signals
18         self.txt_puerto.setText("COM6")
19         self.arduino = None
20         self.btn_accion.clicked.connect(self.accion)
21
22         #control_led = control de actuadores...
23         self.btn_control_led.clicked.connect(self.control_led)
24         self.btn_control_led.setText("ENVIAR")
25
26         self.segundoPlane = QtCore.QTimer()
27         self.segundoPlane.timeout.connect(self.control)
28
29         # Área de los Slots
30
31 MyApp > control_led() > if not self.arduino == None > if self.arduino.isOpen()

```

ReynaSantos501

```
30 def control_led(self):
31     if not self.arduino == None:
32         if self.arduino.isOpen():
33             #1. Leer desde el arduino la informacion de los sensores (progra
34
35             #2. Procesar los datos para tomar una decision
36
37             #3. Enviar la decision tomada a arduino...
38             a = 1
39             b = 10
40             c = 220
41             #PWM = [0 - 255]
42             actuador1 = str(a)
43             actuador2 = str(b)
44             actuador3 = str(c)
45
46             actuador1 = "0" * (3-len(actuador1)) + actuador1
47             actuador2 = "0" * (3-len(actuador2)) + actuador2
48             actuador3 = "0" * (3-len(actuador3)) + actuador3
49
50             cadena = "E" + actuador1 + "R" + \
51                 actuador2 + "R" + actuador3 + "J"
52             print(cadena)
53             # E1R10R220J --> E001R010R220J
54
55             self.arduino.write(cadena.encode())
56
```

ReynaSantos501

```
def control(self):
    try:
        if self.arduino.inWaiting(): #Serial.available()>0
            valor = self.arduino.readline().decode()
            valor = valor.replace("\r","")
            valor = valor.replace("\n","")
            if valor != "": #Solo agregamos cadenas que no esten vacias
                print(valor) #consola

                if valor[0] == "E" and valor[-1] == "C": #Cambiar a J o C
                    #valor.split()
                    subCadena = valor[1:len(valor)-1]
                    datos = subCadena.split("R")
                    cadena = datos[0] + " A " + datos[1] + " A " + datos[2]
                    #pass

                    self.lw_datos.addItem(cadena) #string
                    self.lw_datos.setCurrentRow(self.lw_datos.count()-1)
    except Exception as error:
        print(error)
```

ReynaSantos501

```
def accion(self):
    try:
        if self.arduino == None:
            puerto = self.txt_puerto.text()
```

```
ReynaSantos501
78 def accion(self):
79     try:
80         if self.arduino == None:
81             puerto = self.txt_puerto.text()
82             self.arduino = conectaX.Serial(puerto, baudrate=9600, timeout=1)
83             self.btn_accion.setText("DESCONECTAR")
84             self.txt_estado.setText("CONECTADO")
85             self.segundoPlane.start(10)
86         elif self.arduino.isOpen():
87             self.segundoPlane.stop()
88             self.arduino.close()
89             self.btn_accion.setText("CONECTAR")
90             self.txt_estado.setText("DESCONECTADO")
91         else:
92             self.arduino.open()
93             self.btn_accion.setText("DESCONECTAR")
94             self.txt_estado.setText("CONECTADO")
95             self.segundoPlane.start(10)
96     except Exception as error:
97         print(error)
98
99
100 if __name__ == "__main__":
101     app = QtWidgets.QApplication(sys.argv)
102     window = MyApp()
103     window.show()
104     sys.exit(app.exec_())
```

```
P_11_SerializacionActuador.py x 50
1 import sys
2
3 import serial as conectaX #para trabajar con Arduino
4
5 from PyQt5 import uic, QtWidgets, QtCore
6
7 qtCreatorFile = "P_11_GUI_Python_Arduino.ui" # Nombre del archivo aquí.
8
9 Ui_MainWindow, QtBaseClass = uic.loadUiType(qtCreatorFile)
10
11
12 Emmanuel Alejandro Ruiz Garcia
13 class MyApp(QtWidgets.QMainWindow, Ui_MainWindow):
14     Emmanuel Alejandro Ruiz Garcia
15     def __init__(self):
16         QtWidgets.QMainWindow.__init__(self)
17         Ui_MainWindow.__init__(self)
18         self.setupUi(self)
19
20         #self.txt_puerto.setText("")
21
22         # Área de los Signals
23         self.arduino = None
24         #control led = control dde actuadores...
25         self.btn_accion.clicked.connect(self.accion)
26
27         self.btn_accion.clicked.connect(self.control_led)
28         self.btn_control_led.setText("ENVIAR")
29         self.segundoPlano = QtCore.QTimer()
30         self.segundoPlano.timeout.connect(self.control)
```

```

33 # Área de los Slots
    Emmanuel Alejandro Ruiz Garcia
34 def control_led(self):
35     if not self.arduino == None:
36         if self.arduino.isOpen():
37             #1.- Leer desde el arduino la informacion de los sensores
38             #2.- Procesar los datos para tomar una decision
39             a = 1
40             b = 10
41             c = 220
42             #3.- Enviar la descion tomada a arduino ....
43
44             #PWM = [0-255]
45             actuador1 = str(1)
46             actuador2 = str(10)
47             actuador3 = str(220)
48
49             actuador1 = "0" * (3-len(actuador1)) + actuador1
50             actuador2 = "0" * (3 - len(actuador2)) + actuador2
51             actuador3 = "0" * (3 - len(actuador3)) + actuador3
52
53
54             cadena = "E" + str(actuador1) + "R" + \
55                     str(actuador2) + "R" + str(actuador3) + "J"
56
57             print(cadena)
58             # ER10R220J pasara de esto ---> E001R010R220J
59             self.arduino.write(cadena.encode())
60

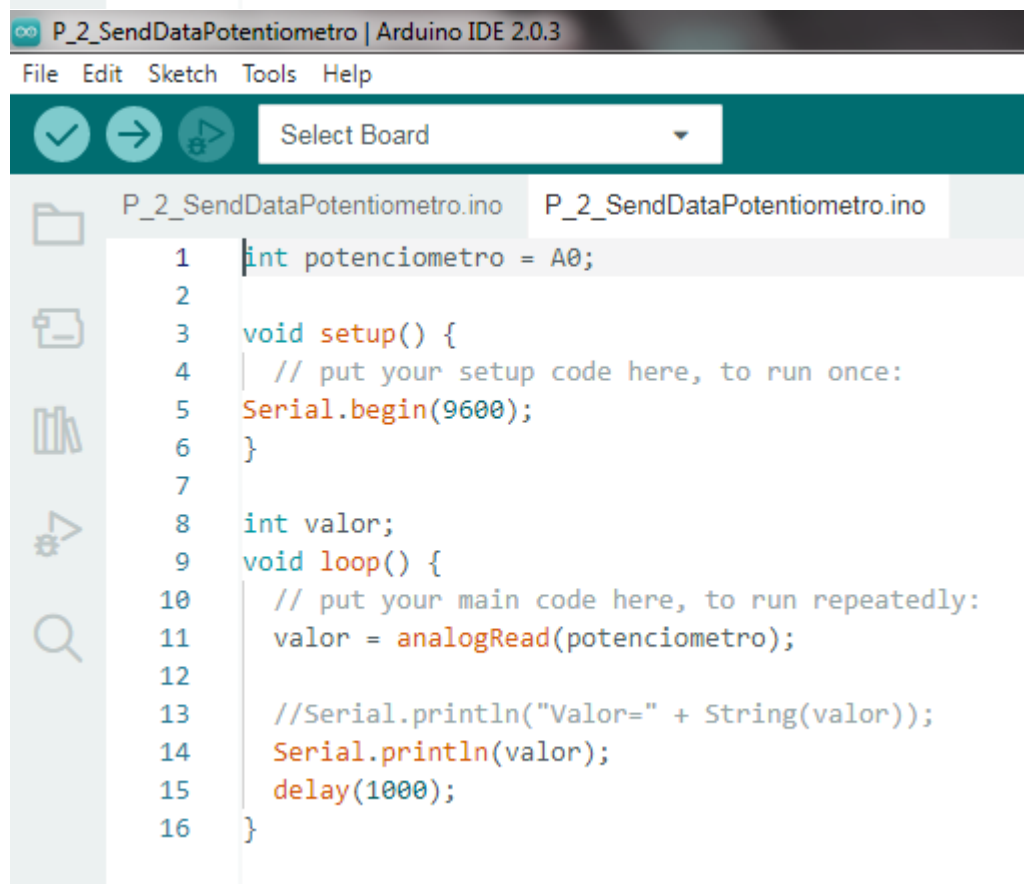
```

Arduino



The screenshot shows the Arduino IDE 2.0.3 interface. The title bar reads "P_1_SendToPython | Arduino IDE 2.0.3". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar contains icons for a checkmark, a right arrow, a play button, and a "Select Board" dropdown menu. The file explorer on the left shows a folder icon and a file icon. The main editor area displays the code for "P_1_SendToPython.ino".

```
1 void setup() {  
2   // put your setup code here, to run once:  
3   Serial.begin(9600);  
4 }  
5  
6  
7 void loop() {  
8   // put your main code here, to run repeatedly:  
9   Serial.println("Mensaje");  
10  delay(1000);  
11 }
```



The screenshot shows the Arduino IDE 2.0.3 interface. The title bar reads "P_2_SendDataPotentiometro | Arduino IDE 2.0.3". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar contains icons for a checkmark, a right arrow, a play button, and a "Select Board" dropdown menu. The file explorer on the left shows a folder icon and a file icon. The main editor area displays the code for "P_2_SendDataPotentiometro.ino".

```
1 int potenciometro = A0;  
2  
3 void setup() {  
4   // put your setup code here, to run once:  
5   Serial.begin(9600);  
6 }  
7  
8 int valor;  
9 void loop() {  
10  // put your main code here, to run repeatedly:  
11  valor = analogRead(potenciometro);  
12  
13  //Serial.println("Valor=" + String(valor));  
14  Serial.println(valor);  
15  delay(1000);  
16 }
```




Select Board



P_3_SendDP_Promedio.ino

P_3_SendDP_Promedio.ino



```
1  int potenciometro = A0;
2
3  void setup() {
4      // put your setup code here, to run once:
5      Serial.begin(9600);
6      //Serial.println();
7  }
8
9  int i,valor;
10 void loop() {
11     // put your main code here, to run repeatedly:
12     for(i=0;i<10; i++){
13         valor += analogRead(potenciometro);
14     }
15     valor /=10;
16
17     //Serial.println("Valor=" + String(valor));
18     Serial.println(valor);
19     delay(1000);
20 }
21
```



Select Board



P_4_SendDP_ValMenor.ino

P_4_SendDP_ValMenor.ino



```
1 int potenciotmetro = A0;
2
3 int totMuestras = 30;
4 void setup() {
5     // put your setup code here, to run once:
6     Serial.begin(9600);
7 }
8 int i , valor , valorMenor = 1024; //9999
9 void loop() {
10    // put your main code here, to run repeatedly:
11    for(i = 0; i < totMuestras; i++){ //-> Teorema del Limite Central
12        valor = analogRead(potenciotmetro);
13        if(valor < valorMenor){
14            valorMenor = valor;
15        }
16    }
17    //valor /= 10;
18    Serial.println(valorMenor);
19    delay(1000);
20
21 }
22
```

```
P_5_SendDP_ValMayor | Arduino IDE 2.0.3
File Edit Sketch Tools Help

Select Board

P_5_SendDP_ValMayor.ino P_5_SendDP_ValMayor.ino

1 int potenciometro = A0;
2 //USAR ARRAY , PROGRAMAR UN METODO DE ORDENAMIENTO , QUICKSORT , BURBUJA ,
3 //TOMAR LOS INDIICES DE EN MEDIO
4 //VALOR MEDIANA = TOMAR ESOS DOS
5 //GUARDAR LAS 30 LECTURAS , OBTENER EL PROMEDIO
6
7
8 /*
9 MODA :
10 SABER EL VALOR QUE MAS SE REPITE. HACER UNA EN EJECUCION X
11
12 GUARDAR LAS 30 ORDENARLAS , DETERMIANR LA FRECUENCIA DE CADA UNO DE LOS NUMEROS , EL QUE APAREZCA MAS VECES ES LA MODA
13
14
15 */
16 int toMuestras = 30;
17 void setup() {
18   // put your setup code here, to run once:
19   Serial.begin(9600);
20 }
21 int i , valor , valorMayor = -1;
22 void loop() {
23   // put your main code here, to run repeatedly:
24   for(i = 0; i <toMuestras; i++){ //-> Teorema del Limite Central
25     valor = analogRead(potenciometro);
26     if(valor>valorMayor){
27       valorMayor = valor;
28     }
29   }
30   //valor /= 10;
31   Serial.println(valorMayor);
32   delay(1000);
33 }
```

```
P_6_Send_GUI_Python | Arduino IDE 2.0.3
File Edit Sketch Tools Help

Select Board

P_6_Send_GUI_Python.ino P_6_Send_GUI_Python.ino

1 void setup() {
2
3   Serial.begin(9600)
4
5 }
6
7 long val = 0
8
9 void loop() {
10   val++;
11   Serial.println(val);
12   delay(100);
13 }
14
```

P_7_ControlLED | Arduino IDE 2.0.3

File Edit Sketch Tools Help

✓

→

⏮

Select Board ▾

P_7_ControlLED.ino

P_7_ControlLED.ino

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

```
int led = 13;

void setup() {
  Serial.begin(9600) ;
  Serial.setTimeout(100);
  //100 miliegunos leyendo la informacion
  pinMode(led,OUTPUT);
}

//long val = 0

void loop() {
  if(Serial.available()>0){
    int v = Serial.readString().toInt();
    digitalWrite(led,v); // 1 0
  }
  delay(10);
}
```

P_8_ControlLED_confirmacion | Arduino IDE 2.0.3

File Edit Sketch Tools Help

✓

→

⚡

Select Board

P_8_ControlLED_confirmacion.ino

P_8_ControlLED_confirmacion.ino

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

```
int led = 13;
void setup() {
  Serial.begin(9600) ;
  Serial.setTimeout(100);
  //100 miliegunos leyendo la informacion
  pinMode(led,OUTPUT);
}

//long val = 0
int estado;

void loop() {
  if(Serial.available()>0){
    estado = Serial.readString().toInt();
    digitalWrite(led,estado); // 1 0
  }

  Serial.println(String(estado));
  delay(10);
}
```



Select Board



P_9_ControlLED_SolicitaConfirmacion.ino

P_9_ControlLED_SolicitaConfirmacion.ino



```
1  int led = 13;
2  void setup() {
3    Serial.begin(9600) ;
4    Serial.setTimeout(100);
5    //100 miliegunos leyendo la informacion
6    pinMode(led,OUTPUT);
7  }
8
9  //long val = 0
10 int estado;
11
12 void loop() {
13   if(Serial.available()>0){
14     estado = Serial.readString().toInt();
15     digitalWrite(led,estado); // 1 0
16     Serial.println()
17   }
18
19   Serial.println(String(estado));
20   delay(10);
21 }
22
```

P_10_LecturaVariosSensores | Arduino IDE 2.0.3

File Edit Sketch Tools Help

✓

→

⚙

Select Board ▼

P_10_LecturaVariosSensores.ino

P_10_LecturaVariosSensores.ino

1

int sensor1 = A0; //pot1

2

int sensor2 = A1; //pot2

3

int sensor3 = A2; //pot3

4

5

6

7

void setup() {

8

Serial.begin(9600) ;

9

//Serial.setTimeout(100);

10

//100 miliegunos leyendo la informacion

11

pinMode(led,OUTPUT);

12

}

13

14

int vP1 , vP2 , vP3;

15

16

void loop() {

17

vP1 = analogRead(sensor1);

18

vP2 = analogRead(sensor2);

19

vP3 = analogRead(sensor3);

20

21

22

Serial.println("E"+ String(vP1) + "G" String(vP2) + "G" String(vP3) + "J");

23

24

25

delay(100);

26

}

27

P_11_EscrituraActuadores | Arduino IDE 2.0.3

File Edit Sketch Tools Help

Select Board

P_11_EscrituraActuadores.ino P_11_EscrituraActuadores.ino

```
1 //Actuadores analogicos
2 int actuador1 = 3;
3 int actuador2 = 5;
4 int actuador3 = 6;
5
6 void setup() {
7   Serial.begin(9600);
8   Serial.setTimeout(10);
9 }
10
11 int vA , vB , vC , cont =0;
12 String cadena;
13 void loop() {
14   if(Serial.available()>0){
15     String c = Serial.readString();
16     //Serial.println(c); //confirmacion de la lectura recibida
17     //E001R010R220J
18
19     if(cadena.length() == 13){ //cadena completa
20       // Serial.println(cadena);
21       if(cadena.charAt(0) == 'E' && cadena.charAt(cadena.length()-1) == 'J'){ //segunda validacion
22         String c = cadena.substring(1,cadena.length()-1);
23         Serial.println("c:" + c + ":D");
24
25         //strtok
26         //charAt y un for
27         //indexOf
28         //en lugar de readString usar read
29         //substring
30         //---->Keystrokes en Python....
31       }
32     }
33   }
34 }
```

Ln 1, Col 1 UTF-8 X No board selected 10:54 p. m. 09/03/2023

P_11_EscrituraActuadores | Arduino IDE 2.0.3

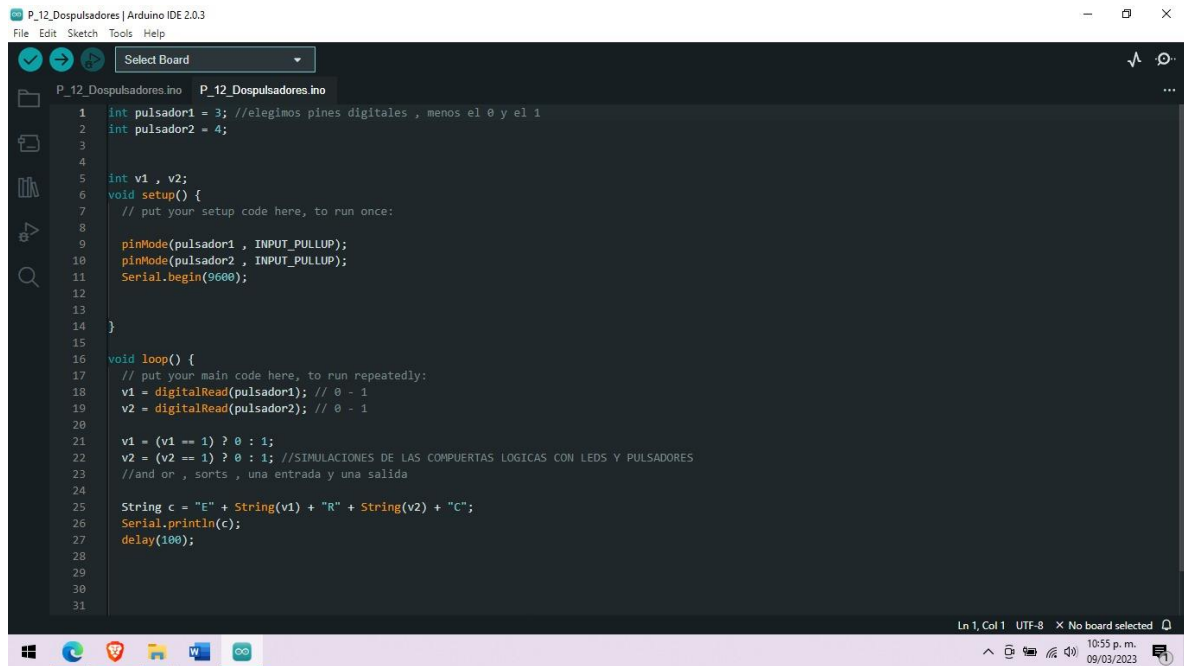
File Edit Sketch Tools Help

Select Board

P_11_EscrituraActuadores.ino P_11_EscrituraActuadores.ino

```
32 //EJEMPLOS DEL FUNCIONAMIENTO DE KEYSTROKES
33
34 //----> Pulsador en su configuracion pullup con Arduino
35 //ES VALIDO SOLO SI SE PUEDE USAR
36 char *cad = c.c_str();
37
38 char *garuco = strtok(cad,"R"); //toquenizador
39
40 //Serial.println(garuco)
41 while(*garuco != NULL){ //INICIO DEL WHILE
42   switch (cont) {
43     case 0:
44       vA = String(garuco).toInt();
45       break;
46     case 1:
47       vB = String(garuco).toInt();
48       break;
49     case 2:
50       vC = String(garuco).toInt();
51       break;
52   }
53   cont++;
54   garuco = strtok(NULL, "R");
55 } //FIN DEL WHILE
56 Serial.println(String(vA) + " " + String(vB) + " " + String(vC));
57
58 }
59
60 }
61 delay(100);
62 }
```

Ln 1, Col 1 UTF-8 X No board selected 10:54 p. m. 09/03/2023



```
1 int pulsador1 = 3; //elegimos pines digitales , menos el 0 y el 1
2 int pulsador2 = 4;
3
4
5 int v1 , v2;
6 void setup() {
7   // put your setup code here, to run once:
8
9   pinMode(pulsador1 , INPUT_PULLUP);
10  pinMode(pulsador2 , INPUT_PULLUP);
11  Serial.begin(9600);
12
13 }
14
15
16 void loop() {
17   // put your main code here, to run repeatedly:
18   v1 = digitalRead(pulsador1); // 0 - 1
19   v2 = digitalRead(pulsador2); // 0 - 1
20
21   v1 = (v1 == 1) ? 0 : 1;
22   v2 = (v2 == 1) ? 0 : 1; //SIMULACIONES DE LAS COMPUERTAS LOGICAS CON LEDS Y PULSADORES
23   //and or , sorts , una entrada y una salida
24
25   String c = "E" + String(v1) + "R" + String(v2) + "C";
26   Serial.println(c);
27   delay(100);
28
29
30
31
```

1.2 Entregas Individuales

NO APLICA EN ESTA UNIDAD

TAREAS E INVESTIGACIONES



TAREAS E INVESTIGACIONES

2.1 Entregas en Equipo

1. Población

Descripción: Investigar el concepto de población.

Desarrollo:

En matemáticas y estadística, la población es un conjunto completo de elementos que comparten una característica común. Puede ser finita o infinita y se utiliza para describir un grupo de personas, objetos o eventos que se desean estudiar. La población es el universo completo del que se obtienen los datos para realizar un análisis estadístico.

Es importante entender que la población no siempre es fácil de definir o medir, por lo que es posible que se tenga que recurrir a una muestra representativa en lugar de analizar la población completa. La muestra representativa es un subconjunto de la población que se utiliza para inferir información sobre la población completa.

Algunos ejemplos de poblaciones incluyen todas las personas en un país, todos los automóviles en una ciudad, todas las estrellas en una galaxia, etc. La población es un concepto clave en estadística y se utiliza para realizar diferentes análisis, como encontrar la media, la desviación estándar, la distribución de probabilidad, etc.

La población también puede ser heterogénea o homogénea. La población heterogénea es aquella que tiene elementos con diferentes características, mientras que la población homogénea es aquella que tiene elementos con características similares.

Además, la población puede ser estática o dinámica. La población estática es aquella en la que los elementos no cambian con el tiempo, mientras que la población dinámica es aquella en la que los elementos pueden agregarse o eliminarse con el tiempo.

En conclusión, la población es un concepto clave en matemáticas y estadística que se utiliza para describir un grupo de elementos que comparten una característica común. La comprensión de la población es esencial para realizar análisis estadísticos precisos y para inferir información sobre un grupo en particular.



Referencias Bibliográficas:

Significados. (2020, 6 julio). *Población*. <https://www.significados.com/poblacion/>

Concepto de Población - Qué es, características, absoluta y relativa. (s. f.). Concepto.

<https://concepto.de/poblacion/>

2. Muestra

Descripción: Investigar el concepto de muestra.

Desarrollo:

En matemáticas y estadística, una muestra es un subconjunto de una población que se utiliza para inferir información sobre la población completa. La muestra representativa es una selección de elementos de la población que se utiliza para analizar y hacer inferencias sobre la población completa.

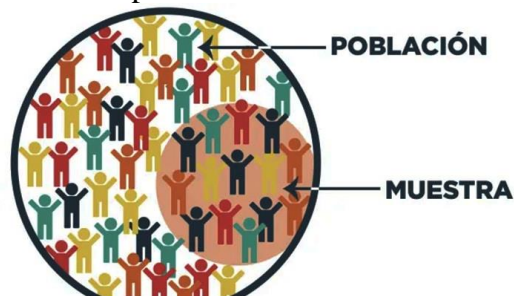
El tamaño de la muestra es un factor importante para considerar en la selección de una muestra representativa. Una muestra grande es más precisa que una muestra pequeña, pero también es más costosa y requiere más tiempo para recopilar y analizar los datos. Por otro lado, una muestra pequeña es menos precisa pero más fácil de manejar y analizar.

Hay diferentes métodos para seleccionar una muestra representativa, como el muestreo aleatorio simple, el muestreo estratificado, el muestreo sistemático, etc. Cada método tiene sus ventajas y desventajas, y es importante elegir el método adecuado en función de las características de la población y los objetivos del análisis.

Una vez seleccionada la muestra, se pueden realizar diferentes análisis estadísticos, como encontrar la media, la desviación estándar, la distribución de probabilidad, etc. Los resultados de estos análisis se utilizan para hacer inferencias sobre la población completa.

Es importante tener en cuenta que los resultados de una muestra pueden no ser representativos de la población completa si la muestra no es representativa o si el tamaño de la muestra es demasiado pequeño. Por lo tanto, es importante seleccionar cuidadosamente una muestra representativa y utilizar un tamaño de muestra adecuado.

En resumen, una muestra es un subconjunto de una población que se utiliza para inferir información sobre la población completa.



Referencias Bibliográficas:

López, J. F. (2022, 24 noviembre). *Muestra estadística*. Economipedia.

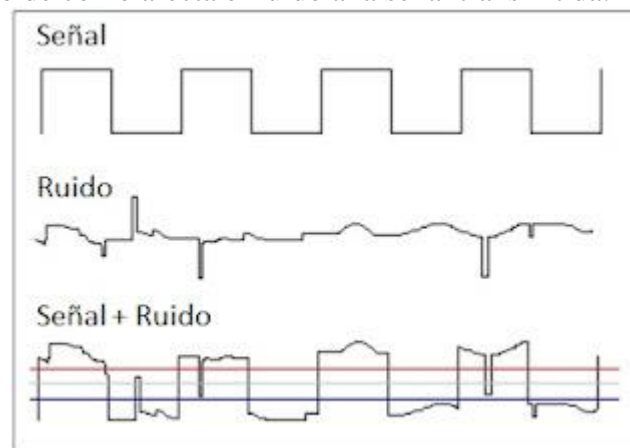
<https://economipedia.com/definiciones/muestra-estadistica.html>

3. Ruido

Descripción: Investigar el concepto de ruido.

Desarrollo:

El término "ruido" se utiliza, generalmente, para señales no deseadas que se han introducido en el sistema de medida e interfieren con la señal a medir y, por tanto, incrementan los errores aleatorios. El ruido en los componentes electrónicos es el resultado de una cantidad mayor o menor de señales eléctricas aleatorias que se acoplan en circuitos en los que no deberían estar, por ejemplo, donde pudieran interrumpir señales de transferencia de información. El ruido se produce tanto en circuitos de señales como de alimentación, pero generalmente, se convierte en un problema cuando se producen en circuitos que manejan señales que representan algún tipo de información. Los circuitos de señales y datos son particularmente vulnerables al ruido, dado que funcionan a altas velocidades y con niveles de baja tensión. Cuanto menor sea la tensión de la señal, menos amplitud de la tensión de ruido se puede tolerar. La relación señal-ruido describe la cantidad de ruido que un circuito puede tolerar antes de que la información sea inválida, es decir, la señal, se vea afectada. En la Fig. 1 se muestra un ejemplo de cómo afecta el ruido a la señal transmitida.



El ruido eléctrico es una señal de interferencia eléctrica no deseada, que se añade o se suma a nuestra señal principal (también denominada “señal útil”), de manera que la puede alterar produciendo efectos que pueden ser más o menos perjudiciales.

Comercios, industrias, hospitales, medios de transporte y hogares se han visto invadidos en los últimos años por equipos eléctricos y electrónicos de todo tipo: teléfonos celulares,

computadoras, equipos de radio y TV, alarmas, sistemas electrónicos del automóvil (ABS, etc.), equipos de electromedicina, etc.

Todos estos equipos procesan energía eléctrica para operar, sin embargo, parte de esa energía se “escapa” de forma incontrolada desde los equipos, bien de forma radiada por el aire o bien de forma conducida por los cables de alimentación.

Si en esa “ruta de escape” alcanzan a otro equipo electrónico y le generan problemas o deficiencias en su funcionamiento, entonces nos encontraremos ante un problema de interferencias electromagnéticas.

En muchas ocasiones esos problemas no son graves, por ejemplo, molestias en receptores de radio y TV debidas a la cercanía de una PC o al utilizar un teléfono celular.

Sin embargo, existen situaciones en las que las consecuencias pueden ser muy graves. Tal es el caso de interferencias electromagnéticas producidas por algunos equipos de comunicación inalámbrica en equipos médicos utilizados para monitorear o mantener la vida de un paciente.

¿Qué Factores Producen Ruido Eléctrico?

La principal fuente de ruido es la red que suministra la energía eléctrica, y lo es porque alrededor de los conductores se produce un campo magnético a la frecuencia de 50 ó 60 Hz. Además, por estos conductores se propagan los parásitos o el ruido producido por otros dispositivos eléctricos o electrónicos.

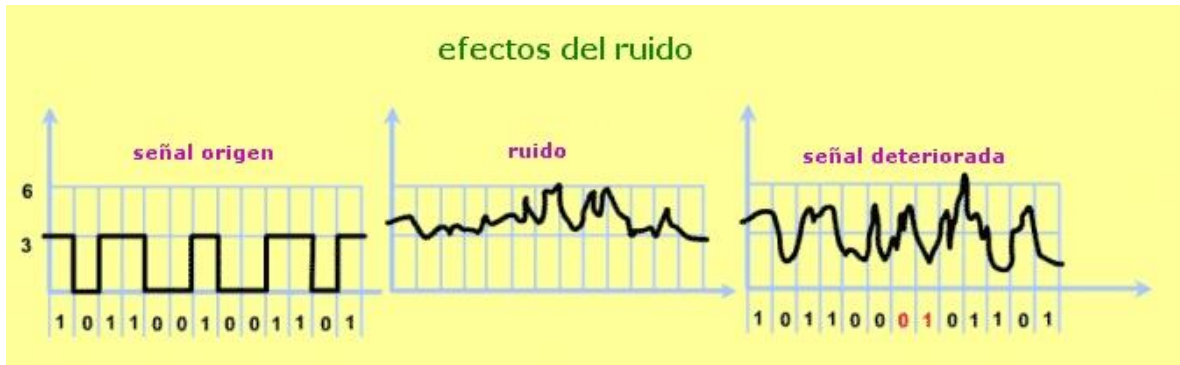
Existen algunas perturbaciones, como las descargas atmosféricas (rayos) que son capaces de actuar desde una gran distancia del lugar en el que se producen, por ejemplo, al caer sobre una línea de alta tensión.

Cuando la señal principal es analógica el ruido será perjudicial en la medida que lo sea su amplitud respecto a la señal principal.

Señal Analógica: Es un tipo de señal generada por algún tipo de fenómeno electromagnético; que es representable por una función matemática continua en la que es variable su amplitud y periodo (representando un dato de información) en función del tiempo. Cuando las señales son digitales, si el ruido no es capaz de producir un cambio de estado, dicho ruido será irrelevante. Sin descartar que el ruido nunca se puede eliminar en su totalidad.

Señal Digital: es un tipo de señal en que cada signo que codifica el contenido de esta puede ser analizado en términos de algunas magnitudes que representan valores discretos, en lugar de valores dentro de un cierto rango.

Para poder atacar de raíz las perturbaciones en la señal útil es necesario conocer las fuentes de ruido.



Referencias Bibliográficas:

U. (s. f.). 7.- *Ruido en componentes electrónicos*.

<https://medind10ma.blogspot.com/2017/07/7-ruido-en-componentes-electronicos.html>

Luis R., J. (2019, 4 octubre). *QUE ES EL RUIDO / Definicion, ejemplos y características*.

247 Tecno. <https://247tecno.com/que-es-el-ruido-definicion-caracteristicas/>

4. Preprocesamiento

Descripción: Investigar el concepto de preprocesamiento.

Desarrollo:

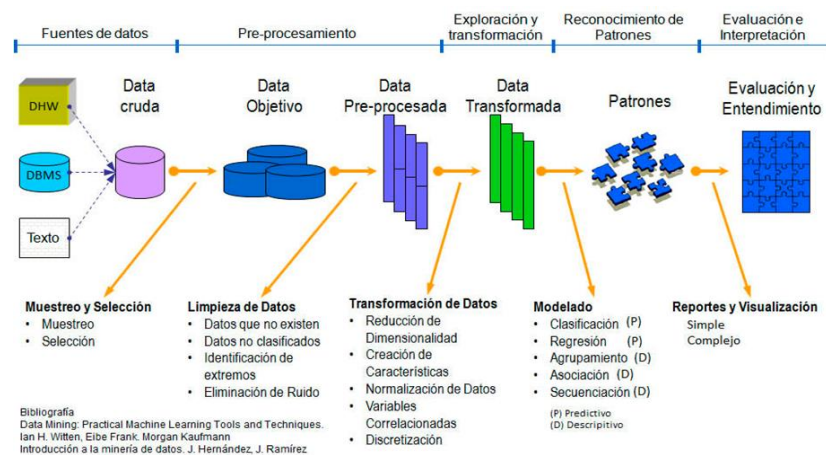
Es la etapa que se realiza antes de cada manipulación y transformación de los datos en un proceso de minería de datos, por lo tanto, es una etapa que podría tener más de una ocurrencia, y no solo al inicio del proyecto como se suele pensar. Las actividades comunes a la etapa de preprocesamiento incluyen la recolección de datos de distintas fuentes, el tratamiento de

cabeceras (header o columnas) y filas (nuestros datos en sí). Con todo esto, se pretende obtener un conjunto de datos de mayor calidad, generalmente más pequeño y homogéneo, el cual deberá conducir a una minería de datos de alta calidad.

El preprocesamiento incluye técnicas tales como la reducción de ruido y realce de detalles. El preprocesamiento de datos es un paso preliminar durante el proceso de minería de datos. Se trata de cualquier tipo de procesamiento que se realiza con los datos brutos para transformarlos en datos que tengan formatos que sean más fáciles de utilizar.

La preparación de datos, también conocida como “preprocesamiento”, es el acto de limpiar y consolidar los datos sin procesar antes de utilizarlos para realizar un análisis de negocio. Puede que no sea la tarea más valorada, pero efectuar una preparación de datos minuciosa es un componente clave para un correcto análisis de datos.

Realizar el proceso de validar, limpiar y aumentar correctamente los datos sin procesar es fundamental para obtener insights precisos y significativos a partir de ellos. La validez y el poder de cualquier análisis de negocio dependen de la eficacia de la preparación de datos realizada en las etapas iniciales.



¿Por qué es importante la preparación de datos?

Las decisiones que toman los líderes dependen de los datos que las respaldan. Una preparación de datos cuidadosa y exhaustiva garantiza que los analistas se sientan seguros, tengan una mayor comprensión y hagan mejores preguntas sobre sus datos, lo que hace que sus análisis sean más precisos y significativos. A partir de un análisis de datos más significativo, se obtienen mejores insights y, por supuesto, mejores resultados.

Referencias Bibliográficas:

Stradata, A. (2021, 30 diciembre). *Preprocesamiento de datos: una forma de solucionar problemas antes de que aparezcan*. Aml | Stradata.

<https://aml.stradata.co/2017/03/27/preprocesamiento-de-datos-una-forma-de-solucionar-problemas-antes-de-que-aparezcan/>

Calidad de datos en minería de datos a través del preprocesamiento. (s. f.).

<https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/calidad-de-datos-en-mineria-de-datos-a-traves-del-preprocesamiento>

5. Teorema del límite central

Descripción: Investigar el concepto y ejemplo del teorema del límite central.

Desarrollo:

El teorema central del límite o teorema del límite central indican que, en condiciones muy generales, si S_n es la suma de n variables aleatorias independientes, con media y varianza finitas, entonces la función de distribución de S_n a una distribución normal (también llamada distribución gaussiana, curva de Gauss o campana de Gauss). El TCL afirma que a medida que el tamaño de la muestra se incrementa, la media muestral se acercará a la media de la población. Por tanto, mediante el TCL podemos definir la distribución de la media muestral de una determinada población con una varianza conocida. De manera que la distribución seguirá una distribución normal si el tamaño de la muestra es lo suficientemente grande.

¿Para qué sirve el teorema central de limite general?

1. Permite averiguar la probabilidad de que la media de una muestra concreta esté en un cierto intervalo.
2. Permite calcular la probabilidad de que la suma de los elementos de una muestra esté, a priori, en un cierto intervalo.

$$\sum_{i=1}^n x_i \rightarrow N(n\mu, \sigma\sqrt{n})$$

3. Inferir la media de la población a partir de una muestra.

Principales propiedades del teorema central del límite

El teorema central del límite tiene una serie de propiedades de gran utilidad en el ámbito estadístico y probabilístico. Las principales son:

- Si el tamaño de la muestra es suficientemente grande, la distribución de las medias muestrales seguirá aproximadamente una distribución normal. El TCL considera una muestra como grande cuando el tamaño de esta es superior a 30. Por tanto, si la muestra es superior a 30, la media muestral tendrá una función de distribución próxima a una normal. Y esto se cumple independientemente de la forma de la distribución con la que estamos trabajando.
- La media poblacional y la media muestral serán iguales. Es decir, la media de la distribución de todas las medias muestrales será igual a la media del total de la población.
- La varianza de la distribución de las medias muestrales será σ^2/n . Que es la varianza de la población dividido entre el tamaño de la muestra.

Que la distribución de las medias muestrales se parezca a una normal es tremendamente útil. Porque la distribución normal es muy fácil de aplicar para realizar contrastes de hipótesis y construcción de intervalos de confianza. En estadística que una distribución sea normal es bastante importante, dado que muchos estadísticos requieren este tipo de distribución. Además, el TCL nos permitirá hacer inferencia sobre la media poblacional a través de la media muestral.

Ejemplo del teorema central del límite

Imaginemos que queremos analizar las rentabilidades medias históricas del índice S&P 500, que como sabemos, tiene unas 500 compañías dentro del mismo. Pero no tenemos suficiente información como para analizar la totalidad de las 500 compañías del índice. En este caso la rentabilidad media del S&P 500 sería la media poblacional.

Ahora bien, siguiendo al TCL podemos coger una muestra de estas 500 empresas para realizar el análisis. La única limitación que tenemos es que en la muestra tiene que haber más de 30 compañías para que se cumpla el teorema. Entonces imaginemos que cogemos 50 compañías del índice de manera aleatoria y repetimos el proceso varias veces.

Los pasos para seguir el ejemplo serían los siguientes:

Elegimos la muestra de unas 50 compañías y obtenemos la rentabilidad media de la totalidad de la muestra.

De manera continuada seguimos escogiendo 50 compañías y obtenemos la rentabilidad media.

La distribución de todas las rentabilidades medias de todas las muestras escogidas se aproximará a una distribución normal.

Las rentabilidades medias de todas las muestras seleccionadas se aproximarán a la rentabilidad media del total del índice. Tal y como demuestra el teorema Central del Límite. Por tanto, mediante inferencia de la rentabilidad media de la muestra podemos acercarnos a la rentabilidad media del índice.

Referencias Bibliográficas:

colaboradores de Wikipedia. (2022, 26 diciembre). *Teorema del límite central*. Wikipedia, la enciclopedia libre.

https://es.wikipedia.org/wiki/Teorema_del_l%C3%ADmite_central

Abellán, J. L. (2021, 9 febrero). *Teorema central del límite (TCL)*. Economipedia.

<https://economipedia.com/definiciones/teorema-central-del-limite.html>

Teorema Central 10.0 - 404.11 - Not Found. (s. f.). <https://www.revistasden.org/files/8-CAP+8.pdf>

6. Modulo PySerial

Descripción: Investigar el concepto y todo lo relacionado con el tema del módulo PySerial.

Desarrollo:

Una vez que tengamos Python instalado para poder comunicarnos con Arduino necesitamos la librería PySerial, que nos permite emplear de forma sencilla el puerto serie. La librería PySerial está disponible en este enlace <https://github.com/pyserial/pyserial> Descargamos y ejecutamos el instalador, para añadir la librería PySerial a nuestra instalación de Python.

También podemos instalar la librería PySerial directamente desde Python, escribiendo el siguiente comando desde una consola.

```
1. python -m pip install PySerial
```

Para configurar el puerto, solo tienes que seleccionar el COM en Windows o el ttyUSB en Linux. La manera más sencilla de revisar en que puerto está el Arduino es en el programa de Arduino en la selección del puerto, ya que solo se habilitara el o los puertos disponibles. Para la velocidad se tiene que configurar la misma que en el Arduino, en este caso nosotros seleccionamos 115200.

Para la escritura del serial, solo usamos la función write y como parámetro de entrada ponemos la cadena de caracteres donde la codificamos con el metodo .encode('utf-8'). La función retorna los bytes escritos en el puerto, solo que en este ejemplo no estamos usando dicho parámetro.

Para la lectura con PySerial, hacemos uso del método. `readline()`, el cual retorna los caracteres disponibles en el puerto serial de la computadora. Cabe señalar que así como se codifica la cadena para el puerto serial, tenemos que decodificar la misma cadena leída con la función. `decode('utf-8')`.

Por último, como estamos trabajando con un formato json, con la función `if`, revisamos si tenemos el caracter de entrada y de salida `<<»` y `<>`. Posteriormente, borramos los posibles caracteres extras que se puedan introducir, usualmente si se envía el dato desde Arduino y usas el `println` envía el LN y el CR como `\r\n`. Con este comando sacamos un sub string para quitar ese extra.

Finalmente, aplicamos el `json.loads` para convertir una cadena en formato de String a formato Json. Con esto podemos hacer uso de la cadena en su forma json de manera formal.

Código PySerial Python

A continuación, se muestra el código y figura de ejemplo de la etapa de Python PySerial.

```
1  #String json_data = "{\"Sesor_id\":\"3E24R\",\"value\":\" + (String)randNumber + \"}";
2
3  import serial, time, json
4
5  hw_sensor = serial.Serial(port='COM4', baudrate=115200, timeout=1, write_timeout=1)
6
7  if __name__ == '__main__':
8      while True:
9          hw_sensor.write('getValue'.encode('utf-8'))
10         time.sleep(1)
11         try:
12             raw_string_b = hw_sensor.readline()
13             raw_string_s = raw_string_b.decode('utf-8')
14             if(raw_string_s.index("{")>=0 and raw_string_s.index("}")==0):
15                 raw_string_s = raw_string_s[0:raw_string_s.index("}")+1]
16                 raw_string_j = json.loads(raw_string_s)
17                 print(raw_string_j)
18                 print(raw_string_j["Sensor_id"])
19                 print(raw_string_j["Value"])
20             else:
21                 print("error/ no } found.")
22         except:
23             print("Exception occurred, something wrong...")
24         hw_sensor.close()
```

```
11         try:
12             raw_string_b = hw_sensor.readline()
13             raw_string_s = raw_string_b.decode('utf-8')
14             if(raw_string_s.index("{")>=0 and raw_string_s.index("}")==0):
15                 raw_string_s = raw_string_s[0:raw_string_s.index("}")+1]
16
17         { 'Sensor_id': '3E24R', 'Value': 15 }
18         3E24R
19         15
20         { 'Sensor_id': '3E24R', 'Value': 32 }
21         3E24R
22         32
23         { 'Sensor_id': '3E24R', 'Value': 19 }
```

Referencias Bibliográficas:

Torres, H. (2021, 28 enero). *PySerial Python Arduino comunicación serial*. HeTPro-Tutoriales. <https://hetpro-store.com/TUTORIALES/pyserial-python-arduino-comunicacion-serial/>

L. (2017, 30 octubre). *Controlar Arduino con Python y la librería PySerial*. Luis Llamas. <https://www.luisllamas.es/controlar-arduino-con-python-y-la-libreria-pyserial/>

Python instalar módulo serial - programador clic. (s. f.).

<https://programmerclick.com/article/42921672706/>

7. Tratamiento del ruido de sensores

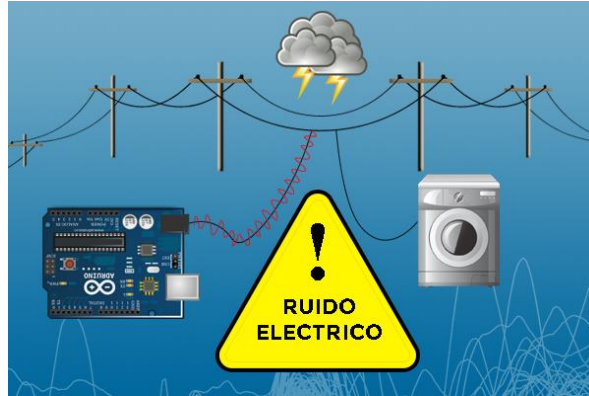
Descripción: Investigar el concepto del tratamiento del ruido de sensores.

Desarrollo:

El ruido en los sensores puede ser un problema importante para la precisión y la fiabilidad de los datos recogidos. Es necesario tratar el ruido en los sensores para garantizar que la información sea precisa y confiable. Hay varios métodos para tratar el ruido en los sensores, incluyendo filtros, muestreo y promedio, reducción de ruido electrónico y calibración.

1. Filtros: existen diferentes tipos de filtros que pueden aplicarse para reducir el ruido en los sensores, como filtros de media móvil, filtros de Kalman, etc.
2. Muestreo y promedio: una forma efectiva de reducir el ruido en los sensores es promediando varias lecturas consecutivas y utilizando el promedio como una representación más precisa de la señal real.
3. Reducción de ruido electrónico: los circuitos electrónicos que se utilizan en los sensores pueden ser optimizados para reducir la cantidad de ruido que se introduce en la señal.
4. Calibración: la calibración regular de los sensores puede ayudar a corregir cualquier desviación en la precisión y reducir el ruido en la señal.

Los filtros son una forma común de tratar el ruido en los sensores. Hay diferentes tipos de filtros disponibles, como filtros de media móvil, filtros de Kalman, entre otros. La selección del filtro adecuado depende de las características específicas del sensor y del sistema en el que se utiliza.



El muestreo y el promedio es otra forma efectiva de reducir el ruido en los sensores. Esto se logra promediando varias lecturas consecutivas y utilizando el promedio como una representación más precisa de la señal real.

La reducción de ruido electrónico también es importante para tratar el ruido en los sensores. Los circuitos electrónicos que se utilizan en los sensores pueden ser optimizados para reducir la cantidad de ruido que se introduce en la señal. Esto puede mejorar la precisión de los datos recogidos.

La calibración es otro método importante para tratar el ruido en los sensores. La calibración regular de los sensores puede ayudar a corregir cualquier desviación en la precisión y reducir el ruido en la señal. La calibración es esencial para garantizar la precisión y la fiabilidad de los datos recogidos por los sensores.

En conclusión, tratar el ruido en los sensores es esencial para garantizar la precisión y la fiabilidad de los datos recogidos. Hay varios métodos disponibles para tratar el ruido en los sensores, incluyendo filtros, muestreo y promedio, reducción de ruido electrónico y calibración. La elección del método adecuado para tratar el ruido en los sensores depende de las características específicas del sensor y del sistema en el que se utiliza, y es importante realizar pruebas y evaluaciones cuidadosas para determinar el método más efectivo en cada caso.

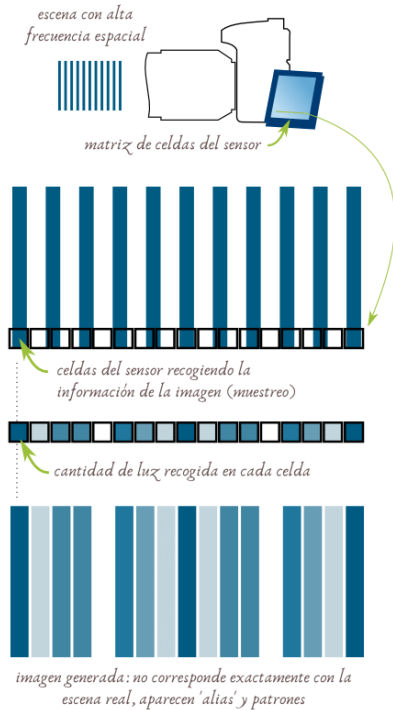
Referencias Bibliográficas:

P. (2021, 1 septiembre). *La Importancia de la Calibración de Sensores de Vibración* -

Predictiva 21. Predictiva21. <https://predictiva21.com/calibracion-sensores-vibracion/>

Métodos de calibración de sensores. (2018, 6 diciembre). HBM.

<https://www.hbm.com/es/4778/metodos-de-calibracion-de-sensores/>



8. Técnicas del filtrado y suavizado para sensores

Descripción: Investigar las técnicas del filtrado y suavizado para sensores.

Desarrollo:

El filtrado y suavizado de datos de sensores es un proceso crucial en la industria y en la investigación en muchos campos, incluyendo la robótica, la automoción, la aeroespacial, la biomedicina y la agricultura, entre otros. La técnica consiste en procesar los datos obtenidos de los sensores para eliminar el ruido y las fluctuaciones que pueden afectar la precisión y la integridad de los datos.

Hay varios métodos diferentes de filtrado y suavizado, cada uno con sus propias fortalezas y debilidades. Algunos de los métodos más comunes incluyen el filtro media móvil, el filtro Gaussiano, el filtro Kalman

y el filtro de partículas. Cada uno de estos métodos utiliza un enfoque diferente para estimar y suavizar los datos, y es importante elegir el método adecuado para una situación dada.

- **Filtro medio móvil:** Este es uno de los métodos más simples y comunes de filtrado y suavizado. Funciona calculando el promedio de un número de muestras consecutivas de los datos del sensor y reemplazando cada punto de datos con ese promedio. Esto suaviza los datos y reduce el ruido.
- **Filtro Gaussiano:** Este método utiliza una función Gaussiana para ponderar las muestras de datos y suavizar los datos. Es más efectivo que el filtro de media móvil en la eliminación del ruido de alta frecuencia, pero puede ser más complicado de implementar.
- **Filtro Kalman:** Este es un método más avanzado que combina un modelo matemático del sistema con la medición del sensor para producir una estimación más precisa de los datos. Es muy útil en situaciones en las que el ruido y las fluctuaciones son importantes.
- **Filtro de partículas:** Este método utiliza un enfoque de "muestreo de partículas" para estimar el estado de un sistema y suavizar los datos. Es muy efectivo en situaciones en las que el modelo del sistema es incierto o desconocido.

Además de la eliminación del ruido, el filtrado y suavizado también puede utilizarse para mejorar la precisión de las mediciones y para detectar patrones y tendencias en los datos. También puede ayudar a eliminar errores y a mejorar la confiabilidad de los datos.

Es importante tener en cuenta que el filtrado y suavizado puede tener un impacto en la velocidad de procesamiento y la complejidad de los algoritmos, y es importante evaluar cuidadosamente el equilibrio entre la precisión y la velocidad en la elección del método de filtrado y suavizado.

En resumen, el filtrado y suavizado es una técnica fundamental para procesar los datos de los sensores y garantizar su precisión y integridad. Hay varios métodos diferentes disponibles, y la elección del método adecuado dependerá de las características específicas de los datos y del sistema de sensores.

Referencias Bibliográficas:

Todo lo que necesitas saber sobre Filtros RC. (s. f.). <https://solectroshop.com/es/blog/todo-lo-que-necesitas-saber-sobre-filtros-rc-n52>

9. Ensayo

La igualación de impedancia es un proceso importante en la transmisión de señales electrónicas para minimizar la reflexión de la señal y maximizar la transferencia de potencia entre dispositivos electrónicos.

La igualación de impedancia puede ser necesaria cuando se transmiten señales de alta frecuencia y se usan cables o líneas de transmisión de cierta longitud. En estos casos, es importante que la impedancia característica de la línea de transmisión y los terminales de los dispositivos coincidan para minimizar la reflexión de la señal y maximizar la transferencia de energía.

Si el procesamiento de la señal es más potente que el procesamiento disponible en el dispositivo receptor puede ocurrir que el receptor no sea capaz de procesar la señal entrante tan rápido como la señal está siendo enviada. Esto puede llevar a una situación en la que el receptor no pueda procesar la señal en tiempo real y, por lo tanto, pueda experimentar una pérdida de datos o una degradación de la calidad de la señal.

Una forma de solucionar este problema es mediante el uso de buffers o memoria intermedia en el receptor. El buffer puede almacenar temporalmente los datos entrantes y permitir que el receptor procese la señal a su propio ritmo. Sin embargo, esto puede introducir cierto retardo en la señal que se está recibiendo, lo que puede no ser deseable en algunas aplicaciones en tiempo real.

En algunos casos, es posible que sea necesario reducir la tasa de transmisión de datos para permitir que el receptor procese la señal de manera efectiva. Esto se puede hacer mediante la implementación de técnicas de control de flujo o mediante la reducción de la tasa de transmisión de datos a un nivel que sea compatible con la capacidad de procesamiento del receptor.

Por lo tanto, en general, es recomendable aplicar la igualación de impedancia cuando se transmiten señales electrónicas entre dispositivos. Sin embargo, hay situaciones en las que la igualación de impedancia puede no ser necesaria o no ser práctica. Por ejemplo, en circuitos de baja frecuencia, la impedancia de los dispositivos y la línea de transmisión puede ser menos crítica y no ser necesaria una igualación precisa. Además, si la longitud de la línea de transmisión es muy corta, la reflexión de la señal puede ser insignificante y la igualación de impedancia puede no ser necesaria.

En general, es importante tener en cuenta la capacidad de procesamiento del dispositivo receptor al diseñar sistemas de transmisión de datos. Es necesario asegurarse de que el receptor tenga suficiente capacidad de procesamiento para manejar la tasa de transmisión de datos esperada y, si es necesario, implementar técnicas de control de flujo o ajustar la tasa de transmisión de datos para evitar problemas de procesamiento lento.

2.2 Entregas Individuales

RESEÑAS DE EXPOSICIONES

¿Que te parecio? PULLUP

El tema estuvo interesante la manera de la configuracion del pulsador usando pull up en Arduino es algo que no conocia ademas de la manera en como se comporta internamente.

¿Que podrias mejorar?

En que el equipo explicara a fondo en como conectarlo de una manera adecuada , ya que lo explicaron de una manera rapida y sin muchos fundamentos.

¿Qué te parecio? PYTHON

La verdad estuvo muy interensante la explicacion de este equipo , el explicar con los archivos de Python que ellos hicieron para aplicarlo en un control lo hizo ver muy facil.

¿Que podrias mejorar?

En si , no mucho , bueno solo que me hubiera gustado como lo pudieron haber implementado en Arduino ya que solamente es mediante en Python.

Pero de ahi en fuera , la explicacion fue muy buena.

PROGRAMAS



PROGRAMAS

3.1 Entregas en Equipo

Enlace al Repositorio: [DEFENDERS-RV/Arduino: Codigos en Arduino \(github.com\)](#)

No.	Nombre del Programa	Estado	Ubicación
1	SEND TO PYTHON	ENTREGADO	En el README del repositorio, en el apartado de PROGRAMAS esta el link directo que lo manda al programa.
2	SEND DATA POTENCIOMETRO	ENTREGADO	En el README del repositorio, en el apartado de PROGRAMAS esta el link directo que lo manda al programa.
3	SEND PROMEDIO	ENTREGADO	En el README del repositorio, en el apartado de PROGRAMAS esta el link directo que lo manda al programa.
4	SEND VALOR MENOR	ENTREGADO	En el README del repositorio, en el apartado de PROGRAMAS esta el link directo que lo manda al programa.
5	SEND VALOR MAYOR	ENTREGADO	En el README del repositorio, en el apartado de PROGRAMAS esta el link directo que lo manda al programa.
6	SEND GUI PYTHON	ENTREGADO	En el README del repositorio, en el apartado de PROGRAMAS esta el link directo que lo manda al programa.
7	CONTROL LED	ENTREGADO	En el README del repositorio, en el

			apartado de PROGRAMAS esta el link directo que lo manda al programa.
8	CONTROL LED CONFIRMACION	ENTREGADO	En el README del repositorio, en el apartado de PROGRAMAS esta el link directo que lo manda al programa.
9	CONTROL LED SOLICITA CONFIRMACION	ENTREGADO	En el README del repositorio, en el apartado de PROGRAMAS esta el link directo que lo manda al programa.
10	LECTURA VARIOS SENSORES	ENTREGADO	En el README del repositorio, en el apartado de PROGRAMAS esta el link directo que lo manda al programa.
11	ESCRITURA ACTUADORES	ENTREGADO	En el README del repositorio, en el apartado de PROGRAMAS esta el link directo que lo manda al programa.
12	DOS PULSADORES	ENTREGADO	En el README del repositorio, en el apartado de PROGRAMAS esta el link directo que lo manda al programa.

3.2 Entregas Individuales

Enlace al Repositorio:

f			
No.	Nombre del Programa	Estado	Ubicación
		{ENTREGADO, INCOMPLETO, NO ENTREGADO}	

PRÁCTICAS



PRÁCTICAS

4.1 Entregas en Equipo

Enlace al Repositorio: [DEFENDERS-RV/Arduino: Codigos en Arduino \(github.com\)](https://github.com/DEFENDERS-RV/Arduino-Codigos-en-Arduino)

No.	Nombre de la Práctica	Estado	Ubicación
1	MODA	ENTREGADO	En el README del repositorio, en el apartado de TAREAS esta el link directo que lo manda al programa.
2	MÉTODO ÚNICO	ENTREGADO	En el README del repositorio, en el apartado de TAREAS esta el link directo que lo manda al programa.
3	MEDIANA	ENTREGADO	En el README del repositorio, en el apartado de TAREAS esta el link directo que lo manda al programa.
4	ASCII LEDS	ENTREGADO	En el README del repositorio, en el apartado de TAREAS esta el link directo que lo manda al programa.
5	JUEGO DEL AHORCADO	ENTREGADO	En el README del repositorio, en el apartado de TAREAS esta el link directo que lo manda al programa.
6	JUEGO DEL GATO	ENTREGADO	En el README del repositorio, en el apartado de TAREAS esta el link directo que lo manda al programa.

4.1.1 Desarrollo de las Prácticas

Práctica 1. MODA

MISMA FRECUENCIA ABSOLUTA MÁXIMA. UNA DISTRIBUCIÓN TRIMODAL DE LOS DATOS ES EN LA QUE ENCONTRAMOS TRES MODAS.

```

1  int potenciometro = A0;
2  const int totMuestras = 30;
3
4  int muestras[totMuestras];
5
6  void setup() {
7      Serial.begin(9600);
8  }
9
10 void loop() {
11     for (int i = 0; i < totMuestras; i++) {
12         muestras[i] = analogRead(potenciometro);
13     }
14
15     // Ordenar las muestras de mayor a menor
16     for (int i = 0; i < totMuestras - 1; i++) {
17         for (int j = i + 1; j < totMuestras; j++) {
18             if (muestras[i] < muestras[j]) {
19                 int temp = muestras[i];
20                 muestras[i] = muestras[j];
21                 muestras[j] = temp;
22             }
23         }
24     }
25
26     // Calcular la moda
27     int moda = muestras[0];
28     int contadorModa = 1;
29     int contador = 1;
30     for (int i = 1; i < totMuestras; i++) {
31         if (muestras[i] == muestras[i - 1]) {
32             contador++;
33             if (contador > contadorModa) {
34                 contadorModa = contador;
35                 moda = muestras[i];
36             }
37         } else {
38             contador = 1;
39         }
40     }
41
42     // Mostrar la moda en el monitor serie
43     Serial.println("Moda: " + String(moda));
44     delay(1000);
45 }

```

Práctica 2. METODO UNICO


```

1 //Actuadores analógicos
2 int actuador1 = 3; // Asigna el pin 3 al primer actuador
3 int actuador2 = 5; // Asigna el pin 5 al segundo actuador
4 int actuador3 = 6; // Asigna el pin 6 al tercer actuador
5
6 void setup() {
7     Serial.begin(9600); // Inicia la comunicación serial con una velocidad de transmisión de 9600 baudios
8     Serial.setTimeout(10); // Configura un tiempo de espera de 10 milisegundos para la lectura serial
9     pinMode(actuador1, OUTPUT); // Configura el primer actuador como salida
10    pinMode(actuador2, OUTPUT); // Configura el segundo actuador como salida
11    pinMode(actuador3, OUTPUT); // Configura el tercer actuador como salida
12 }
13
14 int vA1, vA2, vA3; // Variables para almacenar los valores de voltaje de los actuadores
15
16 void loop() {
17     if (Serial.available() > 0) { // Verifica si hay datos disponibles en la entrada serial
18         String c = Serial.readString(); // Lee la cadena recibida desde la entrada serial y la guarda en la variable c
19         Serial.println(c); // Imprime la cadena leída para confirmar que se recibió correctamente
20
21         if (c.length() == 13) { // Verifica si la cadena recibida tiene una longitud de 13 caracteres
22             Serial.println(c.length()); // Imprime la longitud de la cadena para verificar
23
24             //E001R010R220J
25             if (c.charAt(0) == 'E' && c.charAt(c.length() - 1) == 'J') { // Verifica si la cadena comienza con 'E' y termina con 'J'
26                 c = c.substring(1, c.length() - 1); // Elimina el primer y último carácter de la cadena
27                 c.replace("R", ""); // Elimina todas las ocurrencias de la letra 'R' en la cadena
28
29                 Serial.println(c); // Imprime la cadena resultante para verificar
30
31                 // Separa los valores de voltaje de la cadena
32                 String vA1_str = c.substring(0, 3); // Obtiene los caracteres del 1 al 3 como una subcadena
33                 String vA2_str = c.substring(3, 6); // Obtiene los caracteres del 4 al 6 como una subcadena
34                 String vA3_str = c.substring(6, 9); // Obtiene los caracteres del 7 al 9 como una subcadena
35
36                 // Convierte los valores de voltaje de tipo String a tipo int
37                 vA1 = vA1_str.toInt();
38                 vA2 = vA2_str.toInt();
39                 vA3 = vA3_str.toInt();
40
41                 // Envía los valores de voltaje a los actuadores correspondientes
42                 analogWrite(actuador1, vA1); // Envía el valor de voltaje del primer actuador al pin correspondiente
43                 analogWrite(actuador2, vA2); // Envía el valor de voltaje del segundo actuador al pin correspondiente
44                 analogWrite(actuador3, vA3); // Envía el valor de voltaje del tercer actuador al pin correspondiente
45
46                 // Imprime los valores de voltaje de cada actuador para verificar
47                 Serial.print("Voltaje Actuador 1: ");
48                 Serial.println(vA1);
49                 Serial.print("Voltaje Actuador 2: ");
50                 Serial.println(vA2);
51                 Serial.print("Voltaje Actuador 3: ");
52                 Serial.println(vA3);
53             }
54         }
55     }
56     delay(10); // Retardo de 100 milisegundos antes de leer la entrada serial nuevamente
57 }

```

Práctica 3. MEDIANA

```

1  const int potPin = A0;    // Pin analógico utilizado para leer el potenciómetro
2  int valores[10];          // Array para almacenar los valores leídos del potenciómetro
3  int mediana;              // Variable para almacenar la mediana
4
5  void setup() {
6      Serial.begin(9600);
7  }
8
9  void loop() {
10     // Leer valores del potenciómetro y almacenarlos en el array
11     for (int i = 0; i < 10; i++) {
12         valores[i] = analogRead(potPin);
13         Serial.print(valores[i]); // Imprimir el valor actual
14         Serial.print(",");        // Separador para los valores
15         delay(50);
16     }
17
18     // Ordenar los valores del array
19     for (int i = 0; i < 10; i++) {
20         for (int j = i+1; j < 10; j++) {
21             if (valores[j] < valores[i]) {
22                 int temp = valores[i];
23                 valores[i] = valores[j];
24                 valores[j] = temp;
25             }
26         }
27     }
28
29     // Imprimir los valores ordenados en el monitor serial
30     Serial.print("Valores ordenados: ");
31     for (int i = 0; i < 10; i++) {
32         Serial.print(valores[i]);
33         Serial.print(",");
34     }
35     Serial.println();
36
37     // Calcular la mediana
38     if (10 % 2 == 0) {
39         mediana = (valores[4] + valores[5]) / 2;
40     } else {
41         mediana = valores[5];
42     }
43
44     // Imprimir la mediana en el monitor serial
45     Serial.print("Mediana: ");
46     Serial.println(mediana);
47
48
49     delay(1000); //
50 }

```

Práctica 4. ASCII LEDS

```

1  void setup() {
2      // Configurar los pines de los LED como salidas
3      pinMode(2, OUTPUT);
4      pinMode(3, OUTPUT);
5      pinMode(4, OUTPUT);
6      pinMode(5, OUTPUT);
7      pinMode(6, OUTPUT);
8      pinMode(7, OUTPUT);
9      pinMode(8, OUTPUT);
10     pinMode(9, OUTPUT);
11
12     // Iniciar la comunicación serie a 9600 baudios
13     Serial.begin(9600);
14 }
15
16 void loop() {
17     if (Serial.available() > 0) {
18         String input = Serial.readString();
19         Serial.println(input);
20         // Recorrer cada carácter de la cadena y encender los LEDs correspondientes
21         for (int i = 0; i < input.length(); i++) {
22             int ascii = input.charAt(i);
23             for (int j = 0; j < 8; j++) {
24                 digitalWrite(j + 2, (ascii >> j) & 1);
25             }
26             delay(500); // Esperar medio segundo antes de apagar los LEDs
27             digitalWrite(2, LOW);
28             digitalWrite(3, LOW);
29             digitalWrite(4, LOW);
30             digitalWrite(5, LOW);
31             digitalWrite(6, LOW);
32             digitalWrite(7, LOW);
33             digitalWrite(8, LOW);
34             digitalWrite(9, LOW);
35             delay(500); // Esperar medio segundo antes de continuar con el siguiente carácter
36         }
37     }
38 }

```

Práctica 5. JUEGO DEL AHORCADO

```

1 // Definir la palabra a adivinar
2 String palabra = "ARDUINO";
3
4 // Definir el número de intentos permitidos
5 const int numIntentos = 6;
6
7 // Definir las variables globales
8 int intentos = 0;
9 String palabraAdivinada = "";
10
11 void setup() {
12 // Inicializar la comunicación con el monitor en serie
13 Serial.begin(9600);
14
15 // Inicializar la palabra a adivinar con guiones bajos
16 for (int i = 0; i < palabra.length(); i++) {
17     palabraAdivinada += "_";
18 }
19 }
20
21 void loop() {
22 // Mostrar la palabra adivinada y el número de intentos restantes
23 Serial.println("Palabra a adivinar: " + palabraAdivinada);
24 Serial.println("Intentos restantes: " + String(numIntentos - intentos));
25
26 // Pedir al usuario que ingrese una letra
27 Serial.print("Ingrese una letra: ");
28 while (Serial.available() == 0) {
29 // Esperar a que el usuario ingrese una letra
30 }
31 char letra = Serial.read();
32
33 // Verificar si la letra ingresada está en la palabra a adivinar
34 bool letraCorrecta = false;
35 for (int i = 0; i < palabra.length(); i++) {
36     if (palabra.charAt(i) == letra) {
37         palabraAdivinada.setCharAt(i, letra);
38         letraCorrecta = true;
39     }
40 }
41
42 // Si la letra es incorrecta, incrementar el número de intentos
43 if (!letraCorrecta) {
44     intentos++;
45 }
46
47 // Verificar si el juego ha terminado
48 if (palabraAdivinada == palabra) {
49     Serial.println("¡Felicidades, has ganado!");
50     while (true) {
51         // Esperar aquí para que el juego no se reinicie automáticamente
52     }
53 }
54 else if (intentos == numIntentos) {
55     Serial.println("¡Lo siento, has perdido! La palabra era: " + palabra);
56     while (true) {
57         // Esperar aquí para que el juego no se reinicie automáticamente
58     }
59 }
60 }

```

Práctica 6. JUEGO DEL GATO

```

1  int ledPins[9] = {2, 3, 4, 5, 6, 7, 8, 9, 10}; // Pines para los LEDs
2  int buttonPins[2] = {11, 12}; // Pines para los botones
3
4  int player = 1; // Jugador actual (1 o 2)
5  int board[9] = {0, 0, 0, 0, 0, 0, 0, 0, 0}; // Tablero de juego (0: vacío, 1: jugador 1, 2: jug
6
7  void setup() {
8      // Configuración de los pines
9      for (int i = 0; i < 9; i++) {
10         pinMode(ledPins[i], OUTPUT);
11     }
12     pinMode(buttonPins[0], INPUT_PULLUP);
13     pinMode(buttonPins[1], INPUT_PULLUP);
14 }
15
16 void loop() {
17     // Esperar a que se presione un botón
18     while (digitalRead(buttonPins[0]) == HIGH && digitalRead(buttonPins[1]) == HIGH) {
19         // Mostrar el tablero actual
20         for (int i = 0; i < 9; i++) {
21             if (board[i] == 1) {
22                 digitalWrite(ledPins[i], HIGH); // Jugador 1
23             } else if (board[i] == 2) {
24                 digitalWrite(ledPins[i], LOW); // Jugador 2
25             } else {
26                 digitalWrite(ledPins[i], LOW); // Vacío
27             }
28         }
29         delay(100);
30     }
31
32     // Determinar la posición seleccionada por el jugador
33     int pos = 0;
34     while (digitalRead(buttonPins[0]) == HIGH || digitalRead(buttonPins[1]) == HIGH) {
35         if (digitalRead(buttonPins[0]) == LOW) {
36             pos = 1;
37             break;
38         } else if (digitalRead(buttonPins[1]) == LOW) {
39             pos = 2;
40             break;
41         }
42     }
43     while (digitalRead(buttonPins[0]) == LOW || digitalRead(buttonPins[1]) == LOW) {}
44     delay(100);
45
46     // Actualizar el tablero
47     int index = posToIndex(pos);
48     if (board[index] == 0) {
49         board[index] = player;
50         player = 3 - player; // Cambiar al siguiente jugador
51     }
52 }
53
54 // Función para convertir la posición del botón en un índice del tablero
55 int posToIndex(int pos) {
56     switch (pos) {
57         case 1: return 0;
58         case 2: return 1;
59         case 3: return 2;
60         case 4: return 3;
61         case 5: return 4;
62         case 6: return 5;
63         case 7: return 6;
64         case 8: return 7;
65         case 9: return 8;
66         default: return 0;
67     }
}

```

4.2 Entregas Individuales

Enlace al Repositorio: No aplica en esta unidad

No.	Nombre de la Práctica	Estado	Ubicación
		{ENTREGADO, INCOMPLETO, NO ENTREGADO}	

PROYECTO



PROYECTO

Nombre del Proyecto

Enlace al Repositorio:

Descripción:

¿Qué se tiene que hacer?

Introducción:

Fundamentos teóricos necesarios para realizar la práctica y como estos son necesarios para dar solución a la práctica.

- Incluir Instrucciones o Procedimientos utilizados:

- o Instrucción 1:

- Objetivo
 - Sintaxis
 - Ejemplo

- Incluir información como:

¿Qué es?, ¿Cómo se utiliza?, ¿Para qué nos sirve?, entre otra información.

Desarrollo:

Desarrollo paso a paso de la práctica, explicando en cada paso lo que se hace, para que se hace y que utilidad tiene dentro del programa.

Demostración:

Capturas de pantalla y/o imágenes que demuestren el cumplimiento del objetivo de la práctica. Se deben cada captura y/o imagen.

Conclusiones:

Conclusión general de la práctica de aproximadamente 5-10 renglones.