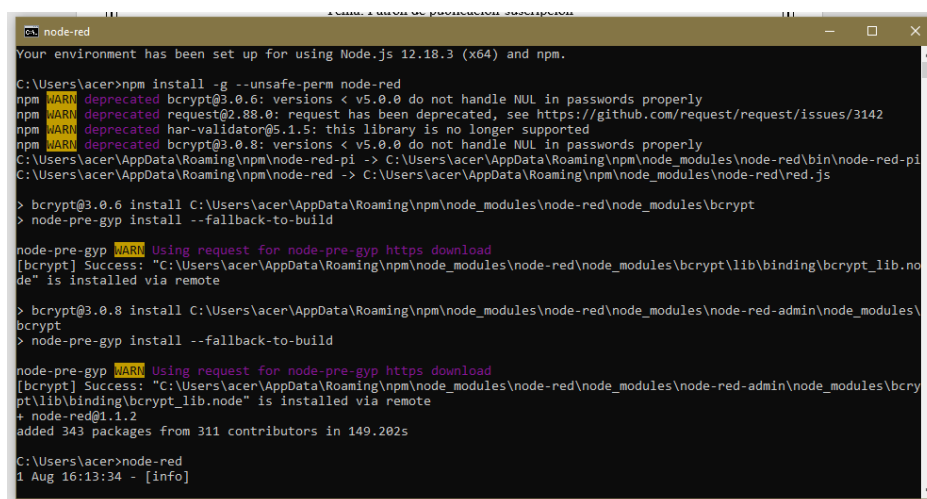


Pasos a seguir en la utilización de Node-red con interfaz HMI

1. Descargar e instalar node-red.
2. Antes de ejecutar el programa debemos descargar el paquete de node-red para ello abrimos nodejs. Command prompt y digitamos lo siguiente “npm install -g --unsafe-perm node-red”, y se comenzara la descarga.



```
C:\Users\acer>npm install -g --unsafe-perm node-red
npm WARN deprecated bcrypt@3.0.6: versions < v5.0.0 do not handle NUL in passwords properly
npm WARN deprecated request@2.88.0: request has been deprecated, see https://github.com/request/request/issues/3142
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated bcrypt@3.0.8: versions < v5.0.0 do not handle NUL in passwords properly
C:\Users\acer\AppData\Roaming\npm\node-red-pl -> C:\Users\acer\AppData\Roaming\npm\node_modules\node-red\bin\node-red-pl
C:\Users\acer\AppData\Roaming\npm\node-red -> C:\Users\acer\AppData\Roaming\npm\node_modules\node-red\red.js

> bcrypt@3.0.6 install C:\Users\acer\AppData\Roaming\npm\node_modules\node-red\node_modules\bcrypt
> node-pre-gyp install --fallback-to-build

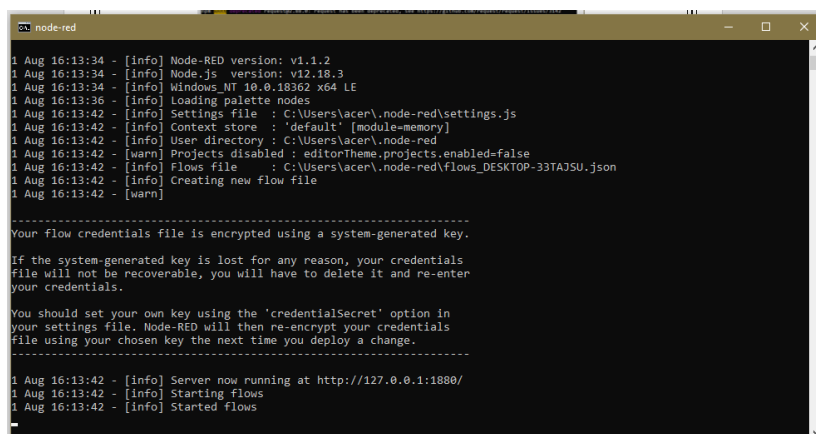
node-pre-gyp WARN Using request for node-pre-gyp https download
[bcrypt] Success: "C:\Users\acer\AppData\Roaming\npm\node_modules\node-red\node_modules\bcrypt\lib\binding\bcrypt_lib.node" is installed via remote

> bcrypt@3.0.8 install C:\Users\acer\AppData\Roaming\npm\node_modules\node-red\node_modules\node-red-admin\node_modules\bcrypt
> node-pre-gyp install --fallback-to-build

node-pre-gyp WARN Using request for node-pre-gyp https download
[bcrypt] Success: "C:\Users\acer\AppData\Roaming\npm\node_modules\node-red\node_modules\node-red-admin\node_modules\bcrypt\lib\binding\bcrypt_lib.node" is installed via remote
+ node-red@1.2.2
added 343 packages from 311 contributors in 149.202s

C:\Users\acer>node-red
1 Aug 16:13:34 - [info]
```

3. Después digitamos “node-red” y nos desplegará una lista, pero lo que nos interesa es lo último donde se nos generó una dirección es la copiamos y la pegamos en nuestro navegador.



```
1 Aug 16:13:34 - [info] Node-RED version: v1.2
1 Aug 16:13:34 - [info] Node.js version: v12.18.3
1 Aug 16:13:34 - [info] Windows_NT 10.0.18362 x64 IE
1 Aug 16:13:36 - [info] Loading palette nodes
1 Aug 16:13:42 - [info] Settings file : C:\Users\acer\.node-red\settings.js
1 Aug 16:13:42 - [info] Context store : 'default' [module=memory]
1 Aug 16:13:42 - [info] User directory : C:\Users\acer\.node-red
1 Aug 16:13:42 - [warn] Projects disabled : editorTheme.projects.enabled=false
1 Aug 16:13:42 - [info] Flows file : C:\Users\acer\.node-red\flows_DESKTOP-33TAJSU.json
1 Aug 16:13:42 - [info] Creating new flow file
1 Aug 16:13:42 - [warn]

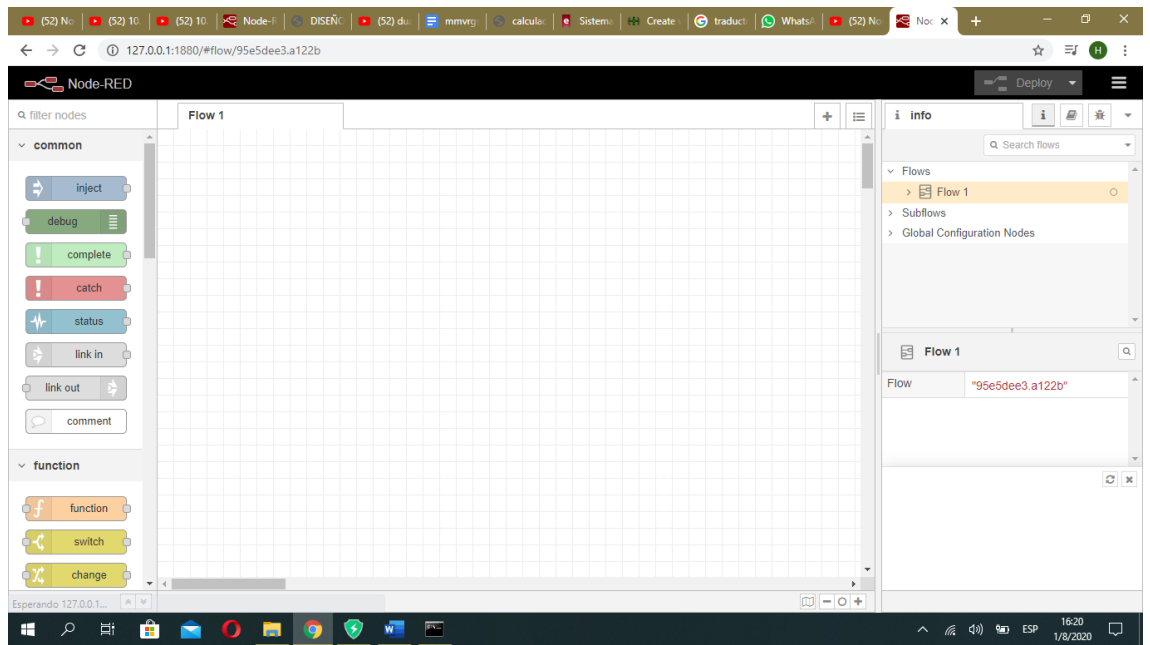
-----
Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
-----

1 Aug 16:13:42 - [info] Server now running at http://127.0.0.1:1880/
1 Aug 16:13:42 - [info] Starting flows
1 Aug 16:13:42 - [info] Started flows
```

4. Una vez ya en nuestro navegador creamos un dashboard donde vamos a crear para poder construir nuestra interfaz HMI



Pasos a seguir en la utilización de la calculadora

1. Nos dirigimos a la página <https://create.withcode.uk/> e iniciamos en primer lugar tenemos que importar la librería que nos permite manejar los GPIO y la librería Math que es la que almacena las funciones. A continuación configuramos las diferentes entradas que tiene la raspberry

{create.withcode.uk}

```
1 # import raspberry pi GPIO module
2 import RPi.GPIO as GPIO
3 import math
4
5 # configuracion de pines de entrada de la raspberry
6 GPIO.setup(3, GPIO.IN)
7 GPIO.setup(5, GPIO.IN)
8 GPIO.setup(7, GPIO.IN)
9 GPIO.setup(8, GPIO.IN)
10 GPIO.setup(10, GPIO.IN)
11 GPIO.setup(11, GPIO.IN)
12 GPIO.setup(12, GPIO.IN)
13 GPIO.setup(13, GPIO.IN)
14 GPIO.setup(15, GPIO.IN)
15 GPIO.setup(18, GPIO.IN)
16 GPIO.setup(19, GPIO.IN)
17 GPIO.setup(21, GPIO.IN)
18 GPIO.setup(22, GPIO.IN)
19 GPIO.setup(23, GPIO.IN)
20 GPIO.setup(38, GPIO.IN)
21 GPIO.setup(40, GPIO.IN)
```

2. Después comenzamos ya a definir nuestra clase y por consiguiente los atributos, los métodos donde vamos a definir las operaciones a ejecutarse, pero en el caso de funciones como **coseno** llamamos a nuestra librería Math y por ejemplo nos que daría así “(math.cos(numero1))”

```
23 #clase
24 class Calculadora:
25     #atributos de clase
26     numero1=0
27     numero2=0
28     #Constructor
29     def __init__(self,numero1,numero2):
30         self.numero1=numero1
31         self.numero2=numero2
32
33     #metodos
34     def sumar(self,numero1,numero2):
35         return (numero1+numero2)
36
37     def restar(self,numero1,numero2):
38         return(numero1-numero2)
39
40     def multiplicar(self,numero1,numero2):
41         return(numero1*numero2)
42
43     def dividir(self,numero1,numero2):
44         return(numero1/numero2)
45
46     def seno(self,numero1):
47         return(math.sin(numero1))
48
49     def coseno(self,numero1):
50         return(math.cos(numero1))
51
52     def tangente(self,numero1):
53         return(math.tan(numero1))
54
55     def rad_grad(self,numero1):
56         return(math.radians(numero1))
```

3. Una vez ya definida nuestras operaciones seguimos a definir nuestro menú de opciones.

```
← → create.withcode.uk ☆ ⌵ H ⋮
73 def potencia(self,numero1,numero2):
74     return(math.pow(numero1,numero2))
75
76
77 def menu(self):
78     opcion=0
79     print("Que desea realizar? (presione un pin en la raspberry): ")
80     print("3) sumar")
81     print("5) restar")
82     print("7) multiplicar")
83     print("8) dividir (numero / numero2)")
84     print("10) sen(x)")
85     print("11) cos(x)")
86     print("12) tan(x)")
87     print("13) convertir Radianes a Grados ")
88     print("15) convertir Grados a Radianes ")
89     print("18) log10(x) (logaritmo base 10 de x)")
90     print("19) log(n)(x) (logaritmo en base n de x, numero = base, numero 2= argumento)")
91     print("21) exponencial e^x ")
92     print("22) raiz cuadrada(x)")
93     print("23) x^y (numero= base, numero 2= exponente)")
```

4. Entonces ya definido nuestro menú utilizamos las entradas GPIO, cabe aclarar que nosotros vamos a definir que entrada vamos a utilizar pero que no debemos olvidar que esas entradas son iguales a las que tiene una raspberry.

```
94
95 while opcion==0:
96     if GPIO.input(3): #sumar
97         opcion=1
98     elif GPIO.input(5): #restar
99         opcion=2
100    elif GPIO.input(7): #mult
101        opcion=3
102    elif GPIO.input(8): #dividir
103        opcion=4
104    elif GPIO.input(19): #logaritmo_n
105        opcion=5
106    elif GPIO.input(23): #potencia
107        opcion=6
108
109    #funciones que necesitan 1 numero
110    elif GPIO.input(10): #sen
111        opcion=7
112    elif GPIO.input(11): #cos
113        opcion=8
114    elif GPIO.input(12): #tan
115        opcion=9
116    elif GPIO.input(13): #grad_rad
117        opcion=10
118    elif GPIO.input(15): #rad_grad
119        opcion=11
120    elif GPIO.input(18): #logaritmo10
121        opcion=12
122    elif GPIO.input(21): #exponencial
123        opcion=13
124    elif GPIO.input(22): #raiz
125        opcion=14
126    return opcion
127
```

5. Lo que sigue es el crear la forma en que se imprimirá por ejemplo si opción = 1 desplegará la suma de esos 2 números y así según se elija en el menú y como nosotros le programamos a los puertos GPIO.

```

131     return men1
132
133 def imprimir(self,num1,num2,resultado,operacion):
134     if operacion == 1:
135         print(num1,"+",num2,"=",resultado)
136     elif operacion == 2:
137         print(num1,"-",num2,"=",resultado)
138     elif operacion == 3:
139         print(num1,"*",num2,"=",resultado)
140     elif operacion == 4:
141         print(num1,"/",num2,"=",resultado)
142     elif operacion == 5:
143         print("log",num1,"(",num2,")","=",resultado)
144     elif operacion == 6:
145         print(num1,"^",num2,"=",resultado)
146     elif operacion == 7:
147         print("sen(",num1,")","=",resultado)
148     elif operacion == 8:
149         print("cos(",num1,")","=",resultado)
150     elif operacion == 9:
151         print("tan(",num1,")","=",resultado)
152     elif operacion == 10:
153         print(num1,"radianes","=",resultado,"grados")
154     elif operacion == 11:
155         print(num1,"grados","=",resultado,"radianes")
156     elif operacion == 12:
157         print("log10(",num1,")","=",resultado)
158     elif operacion == 13:
159         print("e^(",num1,")","=",resultado)
160     elif operacion == 14:
161         print("√(",num1,")","=",resultado)
162
163     else :
164         print("Operacion no establecida")
165

```

6. En este paso vamos a utilizar las entradas GPIO pero las vamos a almacenar en una variables que en este caso será “resultado” pero antes tenemos que que ver que opción es igual a 1 por eso es que se ve asi en la grafica porque se le asigna un número según las opciones del menú.

```

166
167 def operar(self,opcion,num1,num2):
168     resultado=0
169     if opcion==1: #sumar
170         resultado=self.sumar(num1,num2)
171     elif opcion==2: #restar
172         resultado=self.restar(num1,num2)
173     elif opcion==3: #mult
174         resultado=self.multiplicar (num1,num2)
175     elif opcion==4: #dividir
176         resultado=self.dividir(num1,num2)
177     elif opcion==5: #logaritmo_n
178         resultado=self.logaritmo_n(num1,num2)
179     elif opcion==6: #potencia
180         resultado=self.potencia(num1,num2)
181
182     #funciones que necesitan 1 solo argumento
183     elif opcion==7: #sen
184         resultado=self.seno(num1)
185     elif opcion==8: #cos
186         resultado=self.coseno(num1)
187     elif opcion==9: #tan
188         resultado=self.tangente(num1)
189     elif opcion==10: #grad_rad
190         resultado=self.grad_rad(num1)
191     elif opcion==11: #rad_grad
192         resultado=self.rad_grad(num1)
193     elif opcion==12: #logaritmo10
194         resultado=self.logaritmo10(num1)
195     elif opcion==13: #exponencial
196         resultado=self.exponencial(num1)
197     elif opcion==14: #raiz
198         resultado=self.raiz(num1)
199     return resultado
200

```

7. Después creamos una función para que después de cada operación nos pregunte si deseamos continuar o no.

```

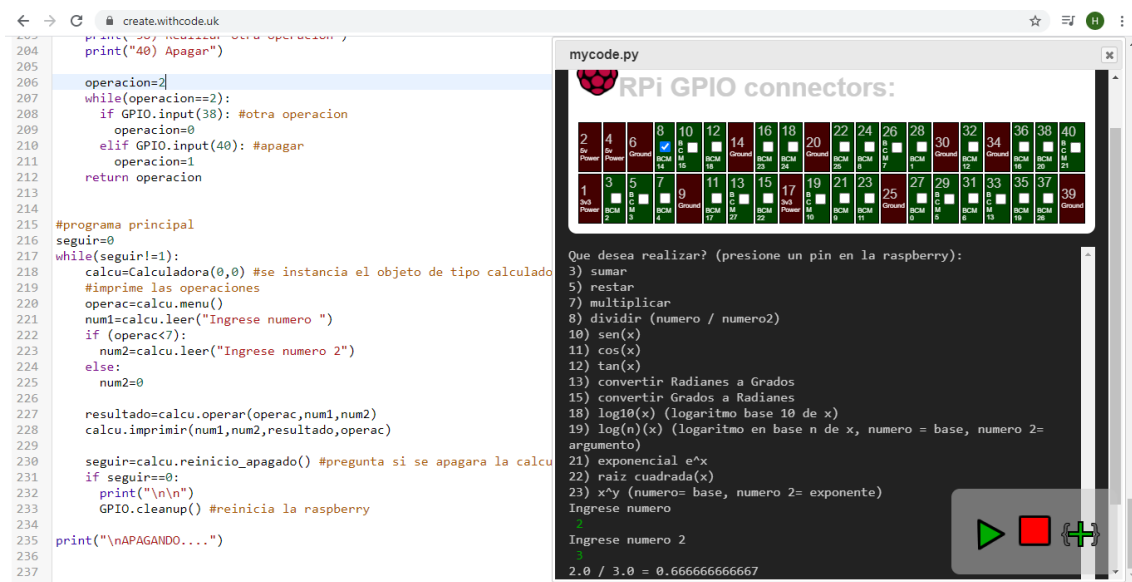
199     return resultado
200
201 def reinicio_apagado(self):
202     print("\nQue desea realizar? (presione un pin en la raspberry): ")
203     print("38) Realizar otra operacion")
204     print("40) Apagar")
205
206     operacion=2
207     while(operacion==2):
208         if GPIO.input(38): #otra operacion
209             operacion=0
210         elif GPIO.input(40): #apagar
211             operacion=1
212     return operacion
213

```

8. En nuestro programa principal ya solo nos queda el ir llamando a cada una de las funciones según se elija en la raspberry.

```
204 print("Otra operacion")
205 print("40) Apagar")
206
207 operacion=2
208 while(operacion==2):
209     if GPIO.input(38): #otra operacion
210         operacion=0
211     elif GPIO.input(40): #apagar
212         operacion=1
213     return operacion
214
215 #programa principal
216 seguir=0
217 while(seguir!=1):
218     calculo=Calculadora(0,0) #se instancia el objeto de tipo calculadora
219     #imprime las operaciones
220     operac=calcu.menu()
221     num1=calcu.leer("Ingrese numero ")
222     if (operac<7):
223         num2=calcu.leer("Ingrese numero 2")
224     else:
225         num2=0
226
227     resultado=calcu.operar(operac,num1,num2)
228     calculo.imprimir(num1,num2,resultado,operac)
229
230     seguir=calcu.reinicio_apagado() #pregunta si se apagara la calculadora
231     if seguir==0:
232         print("\n\n")
233         GPIO.cleanup() #reinicia la raspberry
234
235 print("\nAPAGANDO...")
236
237
```

9. Por último, solo que da hacer las fases de prueba para comprobar si funciona nuestra calculadora científica.



```
← → ↺ create.withcode.uk
204 print("Otra operacion")
205 print("40) Apagar")
206
207 operacion=2
208 while(operacion==2):
209     if GPIO.input(38): #otra operacion
210         operacion=0
211     elif GPIO.input(40): #apagar
212         operacion=1
213     return operacion
214
215 #programa principal
216 seguir=0
217 while(seguir!=1):
218     calculo=Calculadora(0,0) #se instancia el objeto de tipo calculadora
219     #imprime las operaciones
220     operac=calcu.menu()
221     num1=calcu.leer("Ingrese numero ")
222     if (operac<7):
223         num2=calcu.leer("Ingrese numero 2")
224     else:
225         num2=0
226
227     resultado=calcu.operar(operac,num1,num2)
228     calculo.imprimir(num1,num2,resultado,operac)
229
230     seguir=calcu.reinicio_apagado() #pregunta si se apagara la calculadora
231     if seguir==0:
232         print("\n\n")
233         GPIO.cleanup() #reinicia la raspberry
234
235 print("\nAPAGANDO...")
236
237
```

mycode.py

RPi GPIO connectors:

Pin	Function	Pin	Function	Pin	Function	Pin	Function
2	5V	6	BCM 13	10	BCM 19	14	Ground
4	5V	8	BCM 15	12	BCM 23	16	BCM 25
6	Ground	10	BCM 17	14	Ground	18	BCM 27
8	BCM 1	12	BCM 19	16	BCM 25	20	Ground
10	BCM 3	14	Ground	18	BCM 27	22	BCM 29
12	BCM 5	16	BCM 25	20	Ground	24	BCM 31
14	Ground	18	BCM 27	22	BCM 29	26	BCM 33
16	BCM 25	20	Ground	24	BCM 31	28	BCM 35
18	BCM 27	22	BCM 29	26	BCM 33	30	Ground
20	Ground	24	BCM 31	28	BCM 35	32	BCM 37
22	BCM 29	26	BCM 33	30	Ground	34	BCM 39
24	BCM 31	28	BCM 35	32	BCM 37	36	BCM 41
26	BCM 33	30	Ground	34	BCM 39	38	BCM 43
28	BCM 35	32	BCM 37	36	BCM 41	40	BCM 45
30	Ground	34	BCM 39	38	BCM 43		
32	BCM 37	36	BCM 41				
34	BCM 39						
36	BCM 41						
38	BCM 43						
40	BCM 45						

Que desea realizar? (presione un pin en la raspberry):

- 3) sumar
- 5) restar
- 7) multiplicar
- 8) dividir (numero / numero2)
- 10) sen(x)
- 11) cos(x)
- 12) tan(x)
- 13) convertir Radianes a Grados
- 15) convertir Grados a Radianes
- 18) log10(x) (logaritmo base 10 de x)
- 19) log(n)(x) (logaritmo en base n de x, numero = base, numero 2= argumento)
- 21) exponencial e^x
- 22) raiz cuadrada(x)
- 23) x^y (numero= base, numero 2= exponente)

Ingrese numero

Ingrese numero 2

2.0 / 3.0 = 0.666666666667