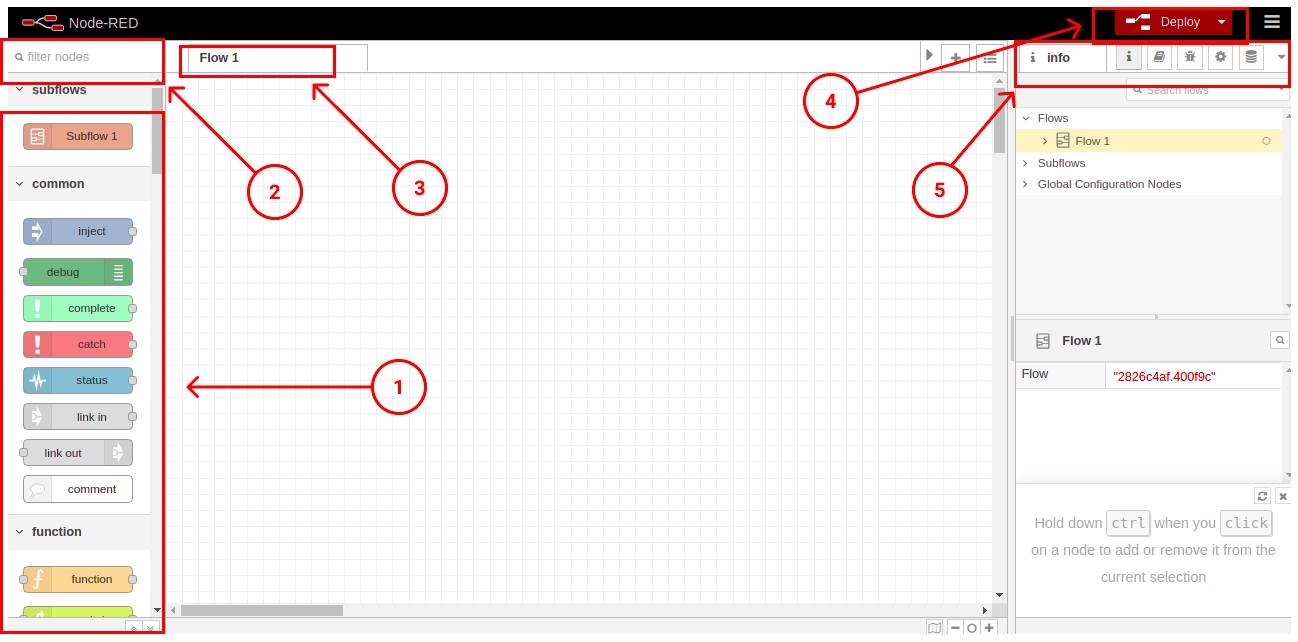
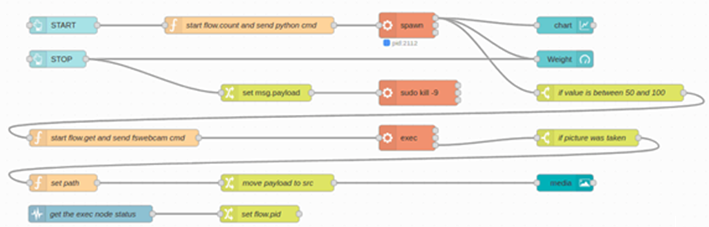
Así pues, vamos a descubrir los aspectos básicos:

1. *node-red-dashboard* o el *node-red-contrib-ui-media*.
2. Por último, en la barra de la derecha donde se muestra la pestaña de información, hay más pestañas importantes como:

* **Information**: para obtener información general de los flujos.
* **Pestaña de ayuda**: que te da información sobre el nodo en el que ha hecho clic.
* **Mensajes de depuración**: es una pestaña muy útil para conocer los errores que se han producido, o para mostrar los mensajes de depuración del nodo.
* **Nodos de configuración**: Muestra los nodos de configuración de los flujos.
* **Panel de control**: Esta pestaña te permite establecer el diseño del tablero, la configuración del sitio y el tema.



* **Nodo Ui\_button**: Añade un botón a la interfaz de usuario. Al hacer clic en el botón se genera un mensaje con msg.payload establecido en el campo Payload. Si no se especifica ningún payload, se utiliza el id del nodo.
* **Nodo de función**: Una función JavaScript que se ejecuta contra los mensajes que recibe el nodo. Los mensajes se pasan como un objeto JavaScript llamado msg. Por convención tendrá una propiedad msg.payload que contiene el cuerpo del mensaje.
* **Nodo Exec**: Ejecuta un comando del sistema y devuelve su salida. El nodo puede ser configurado para esperar hasta que el comando se complete, o para enviar su salida a medida que el comando la genera. El comando que se ejecuta puede ser configurado en el nodo o proporcionado por el mensaje recibido.
* **Nodo Change**: Establece, cambia, elimina o mueve las propiedades de un mensaje, contexto de flujo o contexto global. El nodo puede especificar múltiples reglas que se aplicarán en el orden en que se definan.
* **Nodo Switch**: Encamina los mensajes en función de los valores de sus propiedades o de su posición en la secuencia.
* **Nodo Ui\_chart**: Traza los valores de entrada en un gráfico. Puede ser un gráfico de líneas basado en el tiempo, un gráfico de barras (vertical u horizontal) o un gráfico circular.
* **Nodo Ui\_gauge**: Adds a gauge type widget to the user interface. The msg.payload is searched for a numeric value and is formatted in accordance with the defined Value Format.
* **Nodo Ui\_media**: Muestra archivos multimedia y URLs en el panel de control.
* **Nodo Status**: Informa de los mensajes de estado de otros nodos en la misma pestaña.



**Obtener el valor del peso**

Lo que vamos a hacer es empezar a obtener los valores de un sensor de peso, y cuando la aplicación encuentre el valor que hemos establecido, la cámara USB tomará una foto.

**1.**En primer lugar, vas a añadir dos botones del panel de control: el primero para iniciar la aplicación, y el otro para detenerla.

En el primero, debes crear un Grupo UI y una Pestaña UI para mostrar nuestro dashboard. Una vez hecho, funcionará para todos los nodos del Dashboard, por lo que sólo es necesario una vez. Después de eso, escribirá una etiqueta para ser mostrada, en nuestro caso: START.

|  |  |
| --- | --- |
| Del mismo modo, el botón de parada tendrá la misma configuración; seleccionaremos el grupo y la pestaña donde queremos mostrarlo, escribiremos: STOP como etiqueta y añadiremos un 0 a la carga útil, para que el valor del indicador se ponga a 0 cuando la aplicación se detenga, en lugar de detenerse en el último valor. |  |

Añadir un nodo de función junto al botón de inicio y lo vas a cablear. 

En el nodo de inicio, vas a inicializar una variable de flujo llamada count a 0, que vas a utilizar más adelante cuando nombres las imágenes, y vas a enviar el mensaje con el comando a ejecutar para que la app se inicie.

var count = flow.get('count')||0;

flow.set('count', count);

var newMsg = {payload: "python -u /home/pi/example.py"};

return newMsg;

El nombre del nodo “*start flow.count* y *send python cmd”*.

NODO EXEC PARA TOMAR EL PID

Finalmente, cablea un nodo Exec y edítalo.

* Seleccione la salida: "mientras se ejecuta el comando - *modo spawn*",
* clic en la casilla para añadir el msg.payload.

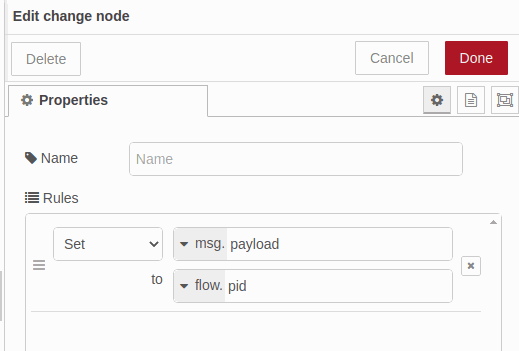
Cuando hay un nodo exec ejecutándose como modo spawn, eso genera un pid del proceso en ejecución, que tendrás que obtener para poder matarlo. Así que eso es lo que vas a hacer ahora.

Añade un *status node*, ve a "*Report status from*" y selecciona "*Selected nodes*".

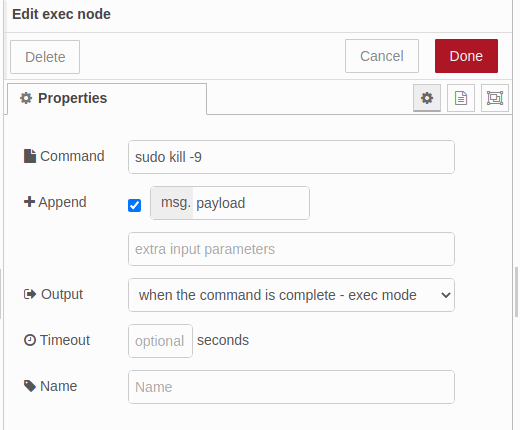
Elige el nodo exec, y haz clic en *Done*.

|  |  |
| --- | --- |
| Después de eso, cablea un nodo Change, y edítalo para establecer el flow.pid como se muestra a continuación: | Node-RED & Raspberry tutorial: How to capture data from sensor |

Finalmente, añade otro nodo de cambio junto al botón de parada y conéctalos. Como hemos establecido el flow.pid en el nodo de cambio anterior, ahora vamos a establecer el msg.payload al flow.pid. Haciendo esto, al pulsar el botón de parada, el msg.payload se enviará a través del nodo.



Ahora el pid es el msg.payload. Añade un nodo exec como modo exec para matar el pid, y edítalo::



 Por el momento te flujo se verá así:

[{"id":"2826c4af.400f9c","type":"tab","label":"Flow 1","disabled":false,"info":""},{"id":"bce0df4f.bc788","type":"ui\_button","z":"2826c4af.400f9c","name":"","group":"c4c1bcc1.49c24","order":16,"width":"7","height":"2","passthru":false,"label":"START ","tooltip":"","color":"","bgcolor":"","icon":"","payload":"","payloadType":"str","topic":"topic","topicType":"msg","x":140,"y":140,"wires":[["882b392c.ab71b8"]]},{"id":"222e70bc.56f6","type":"ui\_button","z":"2826c4af.400f9c","name":"","group":"c4c1bcc1.49c24","order":15,"width":0,"height":0,"passthru":false,"label":"STOP","tooltip":"","color":"","bgcolor":"","icon":"","payload":"0","payloadType":"num","topic":"topic","topicType":"msg","x":130,"y":220,"wires":[["63e42d5b.dee384"]]},{"id":"882b392c.ab71b8","type":"function","z":"2826c4af.400f9c","name":"start flow.count and send python cmd","func":"var count = flow.get('count')||0;\nflow.set('count', count);\n\nvar newMsg = {payload: \"python -u /home/pi/hx711py/example.py\"};\nreturn newMsg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":410,"y":140,"wires":[["9628a2eb.2a5d3"]]},{"id":"2abcf1ce.f1931e","type":"status","z":"2826c4af.400f9c","name":"","scope":["9628a2eb.2a5d3"],"x":140,"y":60,"wires":[["b7fab428.f4fb78"]]},{"id":"9628a2eb.2a5d3","type":"exec","z":"2826c4af.400f9c","command":"","addpay":"payload","append":"","useSpawn":"true","timer":"","oldrc":false,"name":"","x":690,"y":140,"wires":[[],[],[]]},{"id":"b7fab428.f4fb78","type":"change","z":"2826c4af.400f9c","name":"","rules":[{"t":"set","p":"pid","pt":"flow","to":"$number($split(status.text, ':')[1])","tot":"jsonata"}],"action":"","property":"","from":"","to":"","reg":false,"x":410,"y":60,"wires":[[]]},{"id":"63e42d5b.dee384","type":"change","z":"2826c4af.400f9c","name":"","rules":[{"t":"set","p":"payload","pt":"msg","to":"pid","tot":"flow"}],"action":"","property":"","from":"","to":"","reg":false,"x":420,"y":220,"wires":[["46ba8b75.815004"]]},{"id":"46ba8b75.815004","type":"exec","z":"2826c4af.400f9c","command":"sudo kill -9","addpay":"payload","append":"","useSpawn":"false","timer":"","oldrc":false,"name":"","x":690,"y":220,"wires":[[],[],[]]},{"id":"c4c1bcc1.49c24","type":"ui\_group","name":"Group","tab":"cbda5f28.c75ad","order":1,"disp":true,"width":"20","collapse":false},{"id":"cbda5f28.c75ad","type":"ui\_tab","name":"Home","icon":"dashboard","disabled":false,"hidden":false}]

**4.**Ahora, verás los valores de la última 1 hora en un gráfico de líneas, y también en tiempo real en un indicador.

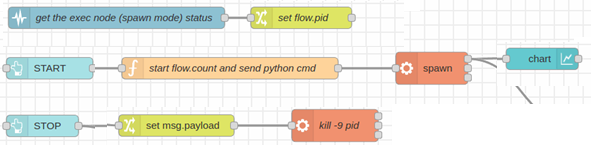
Después, arrastra y suelta un nodo de gráfico y un nodo de indicador, y vamos a editarlos.

En el nodo del gráfico, establece el eje X en la última hora, o en el tiempo que desees registrar, añade la pestaña y el grupo que desees mostrar y haz clic en Listo.

Edita el nodo del indicador eligiendo la misma pestaña y grupo y estableciendo una etiqueta para mostrar como su título, también escriba las unidades. Por último, establezca el valor mínimo y el máximo para fijar el rango:

|  |  |
| --- | --- |
| Node-RED & Raspberry tutorial: How to capture data from sensor | Node-RED & Raspberry tutorial: How to capture data from sensor |

 Por último, conéctalos como se muestra a continuación:



**¡Peso! ¡Fotografíalo!**

Una vez que tienes los valores de nuestra báscula Raspberry y los has mostrado en tu Dashboard, es el momento de hacer algunas fotos.

Para los siguientes pasos es necesario instalar el *node-red-contrib-ui-media*, así que si no lo hiciste aún, por favor ve al último post para saber cómo:

<https://www.industrialshields.com/blog/arduino-industrial-1/post/how-to-take-a-picture-when-a-load-cell-value-is-detected-289>.  
  
**5.**Ahora, vas a añadir un nodo interruptor y establecer si una propiedad está entre 50 y 100 para tomar una foto. Los valores dependen de ti, sólo tienes que elegir las reglas de valor, elegir el campo de número, y añadir el número que quieres presentar. Conecta este nodo al nodo spawn.  
  
**6.** Conectado a la salida del último nodo de cambio, añade un nodo de función para enviar el comando fswebcam y establece el flujo.count para nombrar las imágenes con un contador de la siguiente manera:

var count = flow.get('count');

count++;

flow.set('count', count);

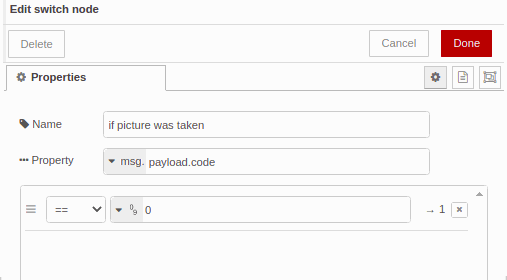
msg.payload = "fswebcam -r 1280x720 --no-banner /home/pi/images/image" + count + ".jpg";

return msg;

Debes añadir tres parámetros al comando *fswebcam*:  
 1. **- r** para establecer la resolución de la foto.  
2.**--no-banner** para omitir el banner de la cámara.  
3. La **ruta**para decir dónde guardar las imágenes, y cómo se van a nombrar.

**7.**El nodo de función enviará un msg.payload, por lo que vas a añadir un nodo exec anexando el msg.payload para ejecutar el comando en tu controlador industrial Raspberry Pi PLC.

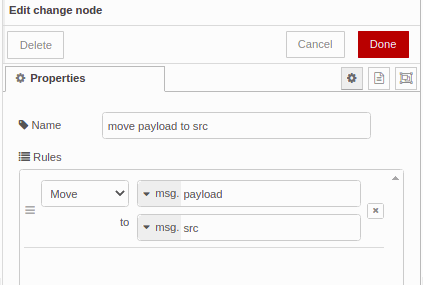
**8.**El modo exec tiene tres salidas. La primera devuelve el stdout, la segunda devuelve el stderr y la última, devuelve el código de retorno. Así que en este caso, conecta la tercera salida, el código de retorno, a un nodo switch para continuar con el flujo si no hubo ningún error.Así que, en el nodo switch, establece la propiedad a msg.payload.code y establece la regla de valor a igual número 0, para estar seguro de que el comando fwwebcam se ejecutó sin errores.

  
  
**9.**Después de eso, conecta un nodo de función para enviar el nombre de la foto que se acaba de tomar, para que se pueda mostrar en el tablero de mando de Node-RED. Una vez editado como se muestra a continuación, conecta un nodo de cambio para mover el msg.payload a msg.src:

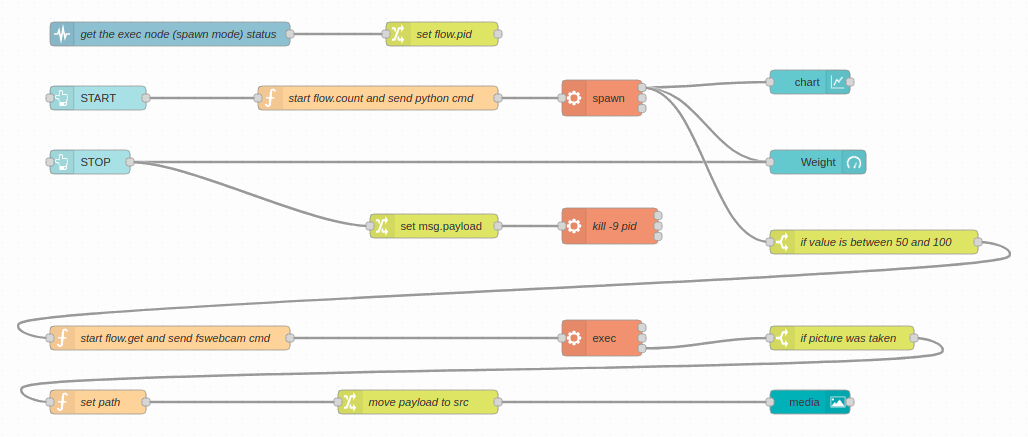
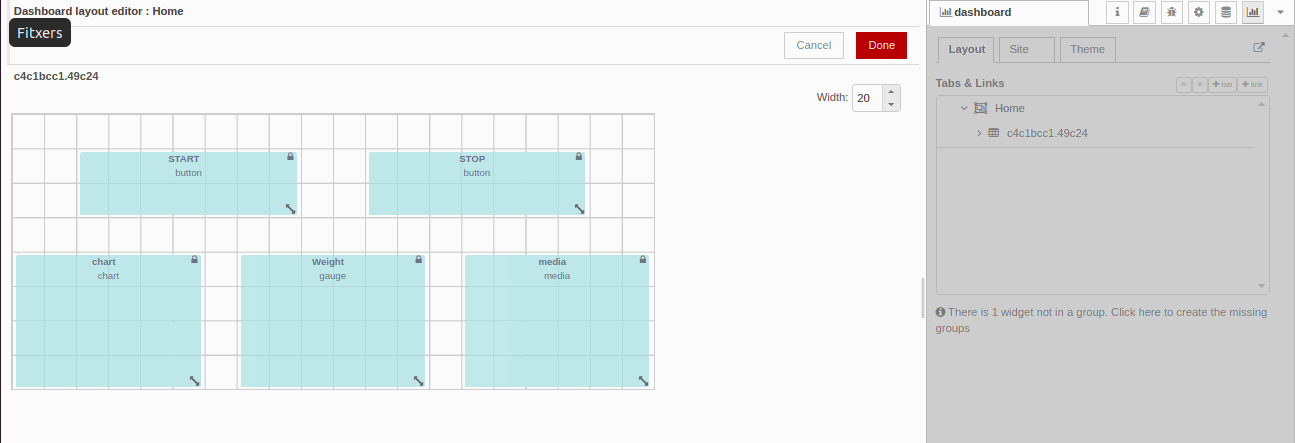
let count = flow.get('count');

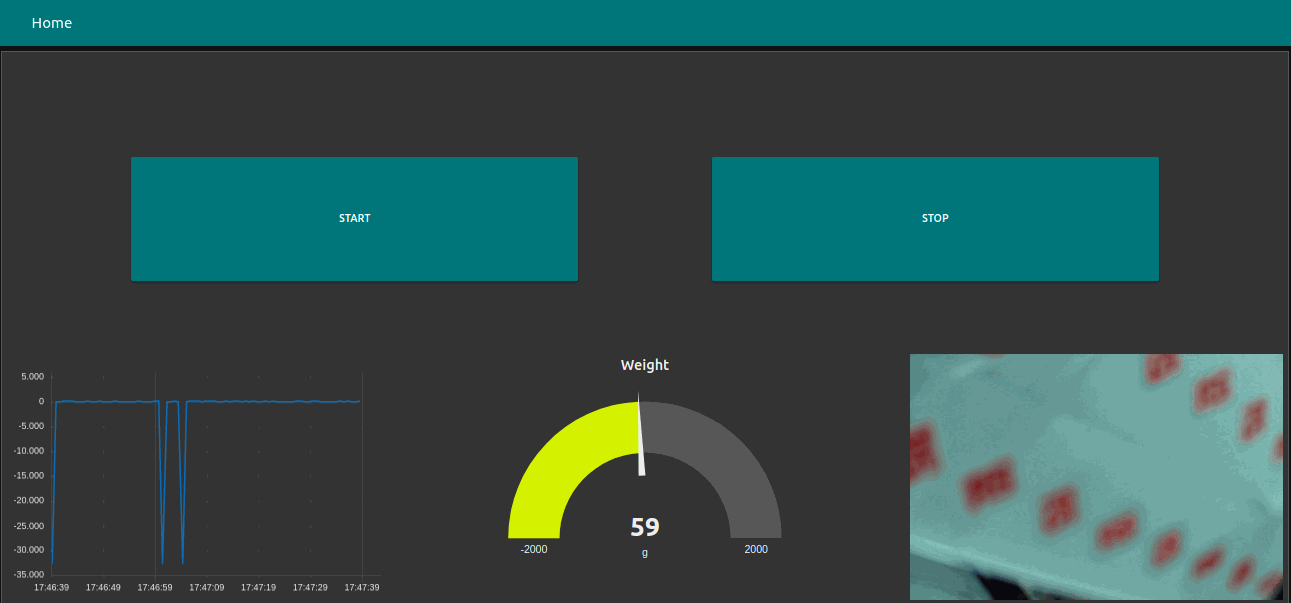
msg.payload = "/image" + count + ".jpg";

return msg;

  
  
**10.**Por último, añada el nodo de medios de comunicación y simplemente añada un grupo a él, o configure el diseño como desee.  
  
Ahora, tu aplicación Node-RED debería ser algo así:

 [{"id":"2826c4af.400f9c","type":"tab","label":"Flow 1","disabled":false,"info":""},{"id":"9b234a13.0256e8","type":"exec","z":"2826c4af.400f9c","command":"","addpay":"payload","append":"","useSpawn":"false","timer":"","oldrc":false,"name":"","x":510,"y":260,"wires":[[],[],["3f27e8b4.d02378"]]},{"id":"ec5481a.4fbf28","type":"exec","z":"2826c4af.400f9c","command":"","addpay":"payload","append":"","useSpawn":"true","timer":"","oldrc":false,"name":"","x":870,"y":60,"wires":[["36307784.3144e8","372e8f7b.f9752","d4e34cd5.f423e"],[],[]]},{"id":"36307784.3144e8","type":"switch","z":"2826c4af.400f9c","name":"if value is between 50 and 100","property":"payload","propertyType":"msg","rules":[{"t":"btwn","v":"50","vt":"num","v2":"100","v2t":"num"}],"checkall":"true","repair":false,"outputs":1,"x":990,"y":160,"wires":[["fb666c7f.c2ff9"]]},{"id":"3f27e8b4.d02378","type":"switch","z":"2826c4af.400f9c","name":"if picture was taken","property":"payload.code","propertyType":"msg","rules":[{"t":"eq","v":"0","vt":"num"}],"checkall":"true","repair":false,"outputs":1,"x":1030,"y":260,"wires":[["7646b60e.83a318"]]},{"id":"7646b60e.83a318","type":"function","z":"2826c4af.400f9c","name":"set path","func":"let count = flow.get('count');\nmsg.payload = \"/image\" + count + \".jpg\";\nreturn msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":140,"y":360,"wires":[["a4078c82.e803a"]]},{"id":"a4078c82.e803a","type":"change","z":"2826c4af.400f9c","name":"move payload to src","rules":[{"t":"move","p":"payload","pt":"msg","to":"src","tot":"msg"}],"action":"","property":"","from":"","to":"","reg":false,"x":560,"y":360,"wires":[["3385b59c.06c81a"]]},{"id":"8771f8be.e44f68","type":"ui\_button","z":"2826c4af.400f9c","name":"","group":"c4c1bcc1.49c24","order":5,"width":7,"height":2,"passthru":false,"label":"STOP LOAD CELL","tooltip":"","color":"","bgcolor":"","icon":"","payload":"0","payloadType":"num","topic":"","topicType":"str","x":170,"y":160,"wires":[["e053ecae.bca31","d4e34cd5.f423e"]]},{"id":"d4e34cd5.f423e","type":"ui\_gauge","z":"2826c4af.400f9c","name":"","group":"c4c1bcc1.49c24","order":13,"width":6,"height":4,"gtype":"gage","title":"Weight","label":"g","format":"{{value}}","min":"-2000","max":"2000","colors":["#00b500","#e6e600","#ca3838"],"seg1":"","seg2":"","x":1070,"y":100,"wires":[]},{"id":"372e8f7b.f9752","type":"ui\_chart","z":"2826c4af.400f9c","name":"","group":"c4c1bcc1.49c24","order":11,"width":6,"height":4,"label":"","chartType":"line","legend":"false","xformat":"HH:mm:ss","interpolate":"linear","nodata":"","dot":false,"ymin":"","ymax":"","removeOlder":1,"removeOlderPoints":"","removeOlderUnit":"3600","cutout":0,"useOneColor":false,"useUTC":false,"colors":["#1f77b4","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"outputs":1,"useDifferentColor":false,"x":1070,"y":60,"wires":[[]]},{"id":"99be7e29.78696","type":"ui\_button","z":"2826c4af.400f9c","name":"","group":"c4c1bcc1.49c24","order":3,"width":7,"height":2,"passthru":false,"label":"START LOAD CELL","tooltip":"","color":"","bgcolor":"","icon":"","payload":"","payloadType":"str","topic":"","topicType":"str","x":180,"y":60,"wires":[["615b1ef6.53963"]]},{"id":"615b1ef6.53963","type":"function","z":"2826c4af.400f9c","name":"start flow.count and send python cmd","func":"var count = flow.get('count')||0;\nflow.set('count', count);\n\nvar newMsg = {payload: \"sudo python -u /home/pi/hx711py/example.py\"};\nreturn newMsg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":610,"y":60,"wires":[["ec5481a.4fbf28"]]},{"id":"3385b59c.06c81a","type":"ui\_media","z":"2826c4af.400f9c","group":"c4c1bcc1.49c24","name":"","width":6,"height":4,"order":15,"category":"","file":"","layout":"expand","showcontrols":true,"loop":true,"onstart":false,"scope":"local","tooltip":"","x":1070,"y":360,"wires":[[]]},{"id":"16de31a2.e4a6de","type":"status","z":"2826c4af.400f9c","name":"get the exec node status","scope":["ec5481a.4fbf28"],"x":190,"y":600,"wires":[["9bb820b3.87fbe"]]},{"id":"9bb820b3.87fbe","type":"change","z":"2826c4af.400f9c","name":"set flow.pid","rules":[{"t":"set","p":"pid","pt":"flow","to":"$number($split(status.text, ':')[1])","tot":"jsonata"}],"action":"","property":"","from":"","to":"","reg":false,"x":410,"y":600,"wires":[["b99c988c.86bf98"]]},{"id":"b99c988c.86bf98","type":"function","z":"2826c4af.400f9c","name":"set kill cmd","func":"let pid = flow.get('pid');\nvar kill = \"kill -9 \" + pid;\nflow.set('kill', kill);\nreturn msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":590,"y":600,"wires":[[]]},{"id":"e053ecae.bca31","type":"function","z":"2826c4af.400f9c","name":"killall python","func":"msg.payload = \"sudo killall python\";\nreturn msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":530,"y":160,"wires":[["ec5481a.4fbf28"]]},{"id":"fb666c7f.c2ff9","type":"function","z":"2826c4af.400f9c","name":"start flow.get and send fswebcam cmd","func":"var count = flow.get('count');\ncount++;\n\nmsg.payload = \"fswebcam -r 1280x720 --no-banner /home/pi/images/image\" + count + \".jpg\";\n\nflow.set('count', count);\n\nreturn msg;\n\n","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":230,"y":260,"wires":[["9b234a13.0256e8"]]},{"id":"5771d86a.220b58","type":"comment","z":"2826c4af.400f9c","name":"In case you want to kill the flow pid and not the python processes, replace the \"killall python\" function node, for the \"killall pid\" function node -->","info":"","x":550,"y":540,"wires":[]},{"id":"b88b67eb.03f068","type":"function","z":"2826c4af.400f9c","name":"killall pid","func":"msg.payload = flow.get('kill');\nreturn msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":1100,"y":540,"wires":[[]]},{"id":"c4c1bcc1.49c24","type":"ui\_group","name":"","tab":"cbda5f28.c75ad","order":1,"disp":true,"width":"20","collapse":false},{"id":"cbda5f28.c75ad","type":"ui\_tab","name":"Home","icon":"dashboard","disabled":false,"hidden":false}]

  
   
  
 **Tips:**Si vas al menú de la derecha, en la pestaña Dashboard, y pasas el ratón por encima de tu pestaña, verás que aparecen tres botones: group, edit y layout. Así, si haces clic en diseño, verás el editor de diseño del Dashboard, donde es posible mostrar tus nodos ui como quieras.  
  


Si ves que no puedes cambiar el tamaño de los widgets, ve a cada nodo del Dashboard, y en la sección de Tamaño, verás que está configurado como *auto*, así que simplemente establece cualquier tamaño manual, vuelve al editor de diseño del Dashboard, donde se aplicarán los cambios.  
  
Finalmente, ve a **http://10.10.10.20:1880/ui/** para comprobar su panel de control Node-RED.  


[How to take a picture when a load cell value is detected | Arduino based PLC Controller for Industrial Applications by Industrial Shields](https://www.industrialshields.com/blog/raspberry-pi-for-industry-26/post/how-to-take-a-picture-when-a-load-cell-value-is-detected-289)

Python code

#! /usr/bin/python2

import time

import sys

EMULATE\_HX711=False

referenceUnit = -1

if not EMULATE\_HX711:

import RPi.GPIO as GPIO

from hx711 import HX711

else:

from emulated\_hx711 import HX711

def cleanAndExit():

print("Cleaning...")

if not EMULATE\_HX711:

GPIO.cleanup()

print("Bye!")

sys.exit()

hx = HX711(25, 2)

hx.set\_reference\_unit(referenceUnit)

hx.reset()

hx.tare()

print("Tare done! Add weight now...")

def func():

while True:

try:

                        val = hx.get\_weight(5)  
                        yield val  
                        hx.power\_down()

hx.power\_up()

time.sleep(0.1)

except (KeyboardInterrupt, SystemExit):

cleanAndExit()

function = func()

for i in function:

print(i)