Develop the Python Script

Date	14 November 2022
Team ID	PNT2022TMID43225
Project Name	Smart waste management system for metropolitan cities

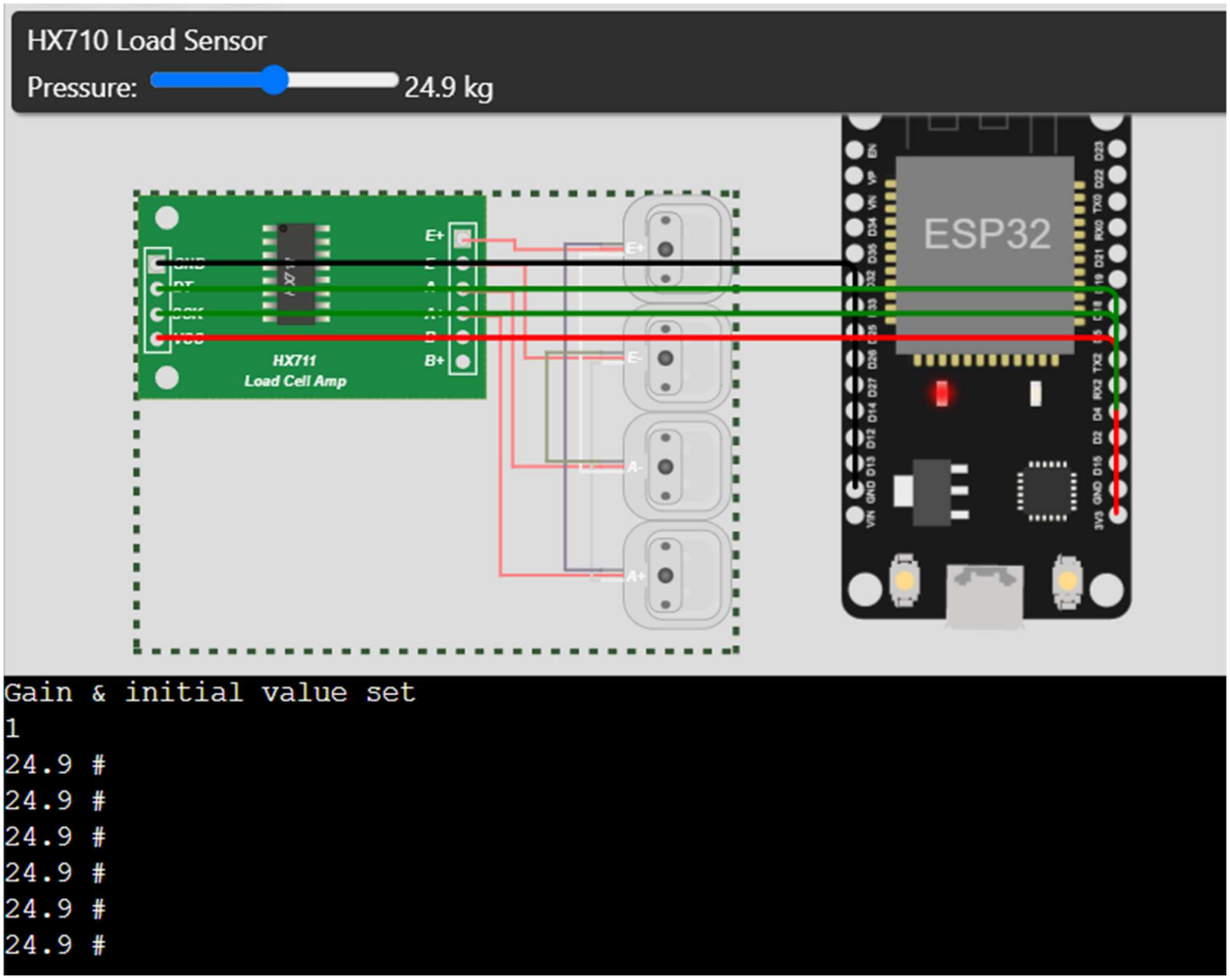
1. Interfacing Load Sensor HX711 with ESP32

WOKWI Code:

```
from hx711 import HX711
hx = HX711(5,4,64)
print(1)
while True:
    hx.tare()
    read = hx.read()
    #average=hx.read_average()
    value=hx.read_average()
    print(value,"#")

from machine import Pin, enable_irq, disable_irq, idle

class HX711:
    def __init__(self, dout, pd_sck, gain=128):

        self.pSCK = Pin(pd_sck , mode=Pin.OUT)
        self.pOUT = Pin(dout, mode=Pin.IN, pull=Pin.PULL_DOWN)
        self.pSCK.value(False)

        self.GAIN = 0
        self.OFFSET = 0
        self.SCALE = 1

        self.time_constant = 0.1
        self.filtered = 0

        self.set_gain(gain);

    def set_gain(self, gain):
        if gain is 128:
            self.GAIN = 1
        elif gain is 64:
            self.GAIN = 3
        elif gain is 32:
            self.GAIN = 2

        self.read()
```

```
        self.filtered = self.read()
        print('Gain & initial value set')

    def is_ready(self):
        return self.pOUT() == 0

    def read(self):
        # wait for the device being ready
        while self.pOUT() == 1:
            idle()

        # shift in data, and gain & channel info
        result = 0
        for j in range(24 + self.GAIN):
            state = disable_irq()
            self.pSCK(True)
            self.pSCK(False)
            enable_irq(state)
            result = (result << 1) | self.pOUT()

        # shift back the extra bits
        result >>= self.GAIN

        # check sign
        if result > 0x7fffff:
            result -= 0x1000000

        return result

    def read_average(self, times=3):
        s = 0
        for i in range(times):
            s += self.read()
        ss=(s/times)/210
        return '%.1f' %(ss)

    def read_lowpass(self):
        self.filtered += self.time_constant * (self.read() - self.filtered)
        return self.filtered

    def get_value(self, times=3):
        return self.read_average(times) - self.OFFSET

    def get_units(self, times=3):
        return self.get_value(times) / self.SCALE

    def tare(self, times=15):
        s = self.read_average(times)
        self.set_offset(s)

    def set_scale(self, scale):
        self.SCALE = scale
```

```
    def set_offset(self, offset):
        self.OFFSET = offset

    def set_time_constant(self, time_constant = None):
        if time_constant is None:
            return self.time_constant
        elif 0 < time_constant < 1.0:
            self.time_constant = time_constant

    def power_down(self):
        self.pSCK.value(False)
        self.pSCK.value(True)

    def power_up(self):
        self.pSCK.value(False)
```



2. Python script to detect the level of the bin:

Using IBM Watson, Node-RED and Node UI:

```
import requests
import json
import ibmiotf.application
import ibmiotf.device
import time
```

```
import random
import sys

# watson device details
organization = "73ffyv"
devicType =  "BIN1"
deviceId = "BIN1ID"
authMethod= "token"
authToken= "123456789"

#generate random values for randomo variables (temperature&humidity)
def myCommandCallback(cmd):
    global a
    print("command recieved is:%s" %cmd.data['command'])
    control=cmd.data['command']
    print(control)

try:
    deviceOptions={"org": organization, "type": devicType,"id": deviceId,"auth-
method":authMethod,"auth-token":authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)
except Exception as e:
    print("Exception while connecting device %s" %str(e))
    sys.exit()

#connect and send a datapoint "temp" with value integer value into the cloud as a type of event for
every 10 seconds
deviceCli.connect()

while True:
    distance= random.randint(10,70)
    loadcell= random.randint(5,15)
    data= {'dist':distance,'load':loadcell}


    if loadcell < 13 and loadcell > 15:
        load = "90 %"
    elif loadcell < 8 and loadcell > 12:
            load = "60 %"
    elif loadcell < 4 and loadcell > 7:
            load = "40 %"
    else:
            load = "0 %"


    if distance < 15:
            dist = 'Risk warning:' 'Garbage level is high, collection time :) 90 %'
    elif distance < 40 and distance >16:
```

```
        dist = 'Risk warning:' 'garbage is above 60%'
    elif distance < 60 and distance > 41:
        dist = 'Risk warning:' '40 %'
    else:
        dist = 'Risk warning:' '17 %'


    if load == "90 %" or distance == "90 %":
        warn  = 'alert :' ' Garbage level is high, collection time :)'
    elif load == "60 %" or distance == "60 %":
        warn = 'alert :' 'garbage is above 60%'
    else :
        warn = 'alert :' 'Levels are low, collection not needed '

    def myOnPublishCallback(lat=11.035081,long=77.014616):
        print("Peelamedu, Coimbatore")
        print("published distance = %s " %distance,"loadcell:%s " %loadcell,"lon = %s " %long,"lat =
%s" %lat)
        print(load)
        print(dist)
        print(warn)

    time.sleep(10)
    success=deviceCli.publishEvent ("IoTSensor","json",warn,qos=0,on_publish=
myOnPublishCallback)
    success=deviceCli.publishEvent ("IoTSensor","json",data,qos=0,on_publish=
myOnPublishCallback)

    if not success:
        print("not connected to ibmiot")

    time.sleep(30)
    deviceCli.commandCallback=myCommandCallback

#disconnect the device
deviceCli.disconnect()
```

The above code is for one of the bins, a similar code is used for other 3 bins but from different locations.