DELIVERY OF SPRINT 2

|  |  |
| --- | --- |
| **Date** | 14 November 2022 |
| **Team ID** | PNT2022TMID52668 |
| **Project Name** | Smart waste management system for metropolitan cities |

**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)

