

DELIVERY OF SPRINT 2

Date	14 November 2022
Team ID	PNT2022TMID41546
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 > 0x7ffffff:
        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)

```

