Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
**«ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра ЭВМ**

ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №1

по дисциплине «Введение в технологии интернета вещей»

Тема: IoT платформы

Выполнил
Студент группы КЭ-120
_____/М.А. Щукин
«____»_____2021 г.

Принял
_____/И.Л. Кафтанников
«____»_____2021 г.

Челябинск 2020

ЗАДАНИЕ

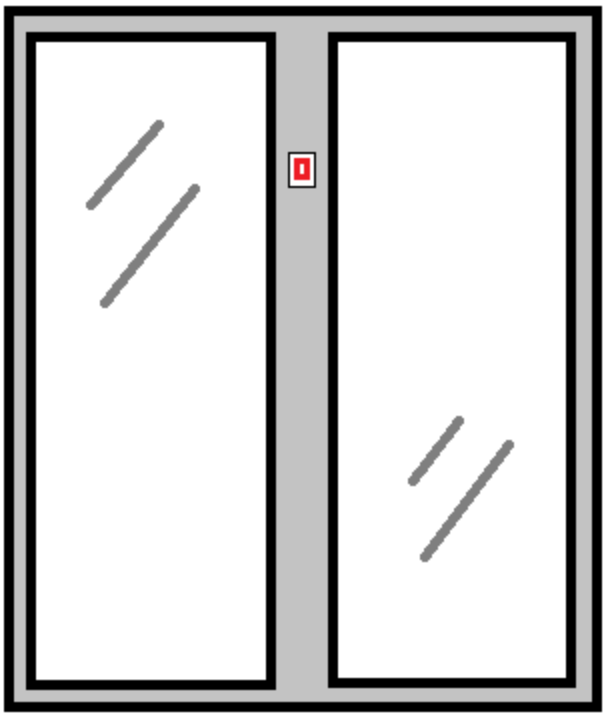
Для выбранных двух датчиков естественной освещённости – Adafruit TSL2561 Digital Ambient Light Sensor описать подключение к Raspberry Pi 4 через I2C интерфейс.

1. ПЕРВЫЙ ВАРИАНТ

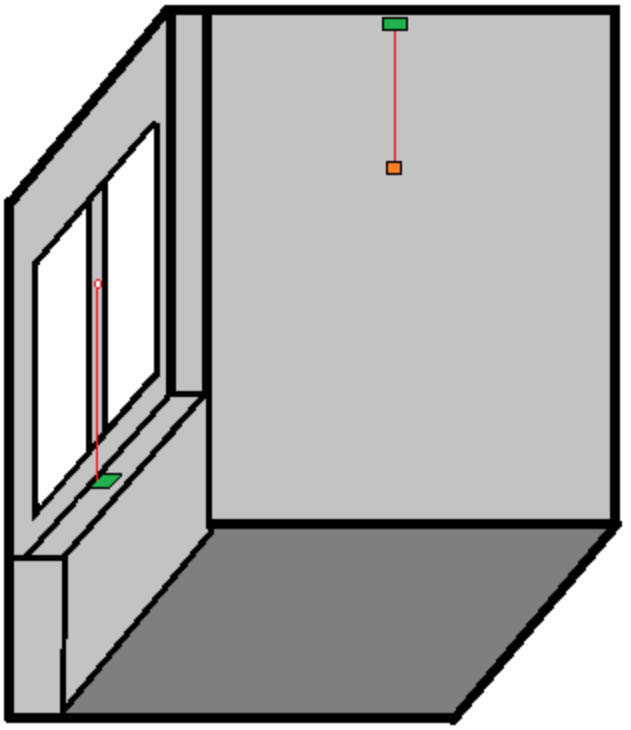
1.1. Описание варианта

В данном варианте каждый датчик подключается к собственной Raspberry Pi как *Slave*. Каждая Raspberry Pi связана с облачным сервером, который и управляет работой всей системы (в данном случае отправляет запрос на сбор информации о текущей освещённости на улице и в помещении).

Расположение и подключение датчиков для этого варианта представлено на рисунках ниже.



Датчик в защитном прозрачном корпусе (красный) снаружи окна



Кабель-канал с проводом, соединяющим RPi (зелёный)
и датчик (красный снаружи) переделай чтобы был внутри чтобы из внутри помещения мерил внешнюю освещённость, а также второй RPI (зелёный) и второй датчик (оранжевый внутри)

Пункты 1.2, 1.3 и 1.4 выполняются для каждой RPi.

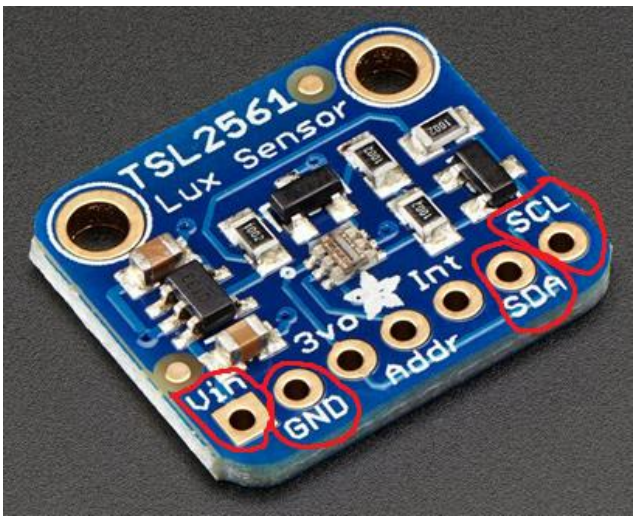
1.2. Физическое подключение

Прежде всего нам понадобится Raspberry Pi 4 с установленной Raspberry Pi OS, подключаться к нему будем через ssh. Третья версия Python должна быть установлена в качестве основной.

Подключение будем осуществлять с помощью male-to-female джамперов.

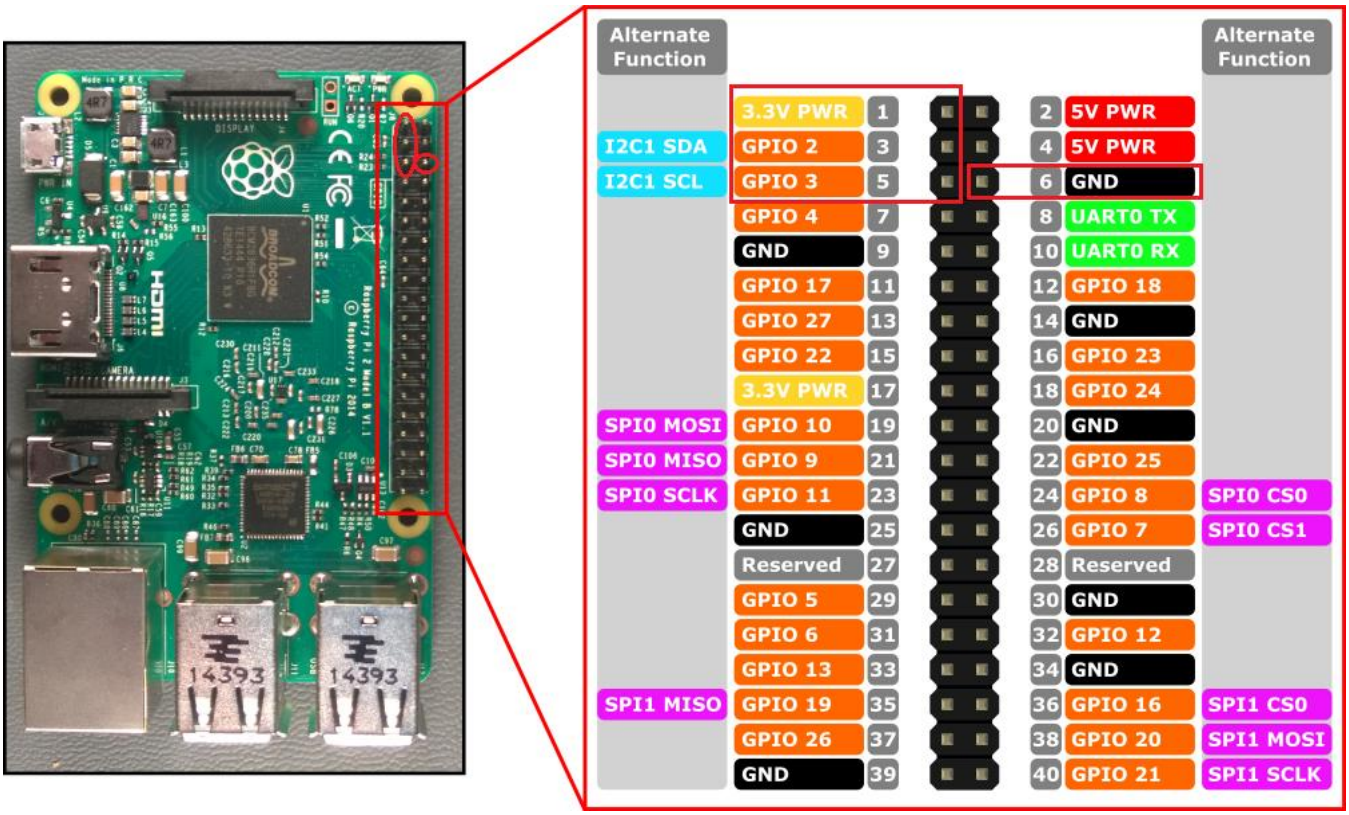
Для начала необходимо соединить:

- Vin – питание 3.3V
- GND – земля
- SDA – data
- SCL – clock



С соответствующими пинами Raspberry Pi:

- 1 – 3.3V
- 6 – GND
- 3 – I2C1 SDA (GPIO 2)
- 5 – I2C1 SCL (GPIO 3)



Также возможна необходимость в экранировании джамперов и кабель-канала. Подойдёт обычная алюминиевая фольга.

1.3. Установка программного обеспечения

Следующий шаг на Raspberry Pi необходимо включить I2C интерфейс. Для этого в терминале прописываем:

```
sudo raspi-config
```

И далее выбираем:
5 Interfacing Options >>> P5 I2C >>> Yes

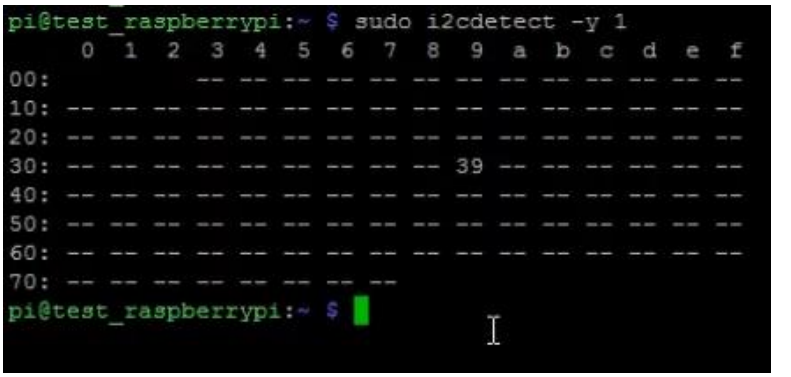
После этого необходимо установить пакет i2c-tools (набор функций для работы Linux с I2C), для этого пропишем в терминале:

```
sudo apt-get install i2c-tools
```

Теперь проверим видит ли Raspberry датчик, для этого пропишем:

```
sudo i2cdetect -y 1
```

В результате мы должны увидеть стандартный адрес датчика 0x39 (если всё подключено как на рисунках выше).



Так как производитель Adafruit предоставляет Python драйвер для упрощения работы с датчиком, воспользуемся им. Исходный код можно найти на GitHub (https://github.com/adafruit/Adafruit_CircuitPython_TSL2561).

Для его работы понадобится установить **Adafruit CircuitPython**. Это небольшая версия Python для микроконтроллеров и микрокомпьютеров имеющая встроенный функционал для работы с датчиками, моторчиками, LED. Она также облегчает работу с I2C, SPI, UART. Для её установки выполним следующие команды в терминале:

```
pip3 install RPI.GPIO
pip3 install adafruit-blinka
```

Также нам понадобится Adafruit CircuitPython BusDevice установим его:
```
pip3 install adafruit-circuitpython-busdevice
```
Теперь можно установить драйвер для датчика:
```
pip3 install adafruit-circuitpython-tsl2561
```

1.4. Работа с датчиком

Можно начать работу с датчиком. Напишем на языке python скрипт для вывода текущей освещённости:

```python
import board
import busio

i2c = busio.I2C(board.SCL, board.SDA)

import adafruit_tsl2561
tsl = adafruit_tsl2561.TSL2561(i2c)
print(tsl.lux)
```

Чтобы запустить скрипт и увидеть результат остаётся только ввести в консоле:

```
python script_name.py
```

1.5. Дополнительно

Если по какой-то причине необходимо более низкоуровневое управление можно выполнять его самостоятельно через регистры см. рисунок с Python кодом ниже. Однако в этом мало смысла так драйвер предоставляет такой же функционал.

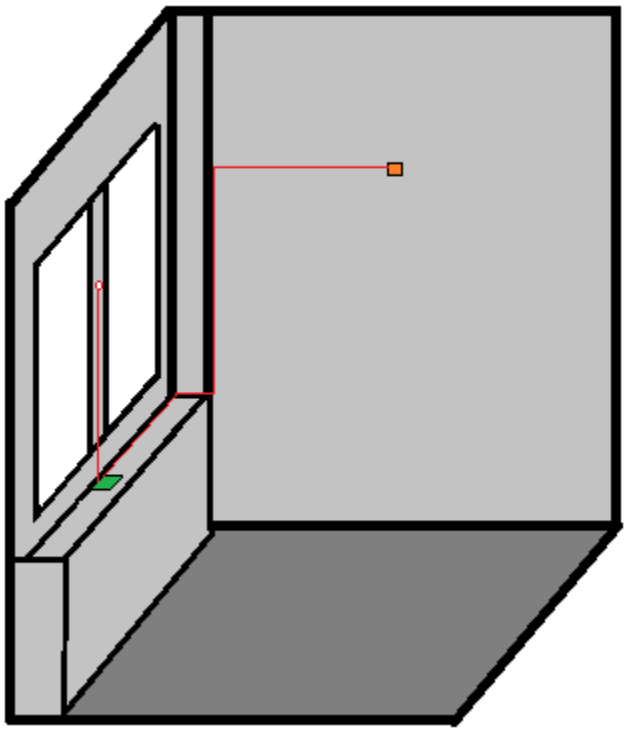
```python
import smbus
import time

bus = smbus.SMBus(1) # I2C шина

# стандартный адрес TSL2561
DEFAULT_ADDRESS = const(0x39)

CONTROL_POWERON = const(0x03)
COMMAND_BIT = const(0x80)

REGISTER_CONTROL = const(0x00)
REGISTER_TIMING = const(0x01)
REGISTER_THRESH_LOW = const(0x02)
REGISTER_DATA_0 = const(0x0C)
REGISTER_DATA_1 = const(0x0E)

# Выбрать командый регистр
bus.write_byte_data(DEFAULT_ADDRESS, REGISTER_CONTROL | COMMAND_BIT, CONTROL_POWERON)

# Выбрать временной регистр, с командым регистром, а также
#   время интеграции = 402 мс (время в течении которого датчик "впитывает" свет
#   его можно понизить до 101мс или 13.7мс, но это уменьшит чувствительность датчика
bus.write_byte_data(DEFAULT_ADDRESS, REGISTER_TIMING | COMMAND_BIT, REGISTER_THRESH_LOW)
time.sleep(0.5)

# LSB - Младший бит
# MSB - Старший бит
# Считать блок данных с REGISTER_DATA_0, с командым регистром, 2 байта
# ch0 LSB, ch0 MSB
data1 = bus.read_i2c_block_data(DEFAULT_ADDRESS, REGISTER_DATA_0 | COMMAND_BIT, 2)

#  Считать блок данных с REGISTER_DATA_1, с командым регистром, 2 байта
# ch1 LSB, ch1 MSB
data2 = bus.read_i2c_block_data(DEFAULT_ADDRESS, REGISTER_DATA_1 | COMMAND_BIT, 2)

# Конвертируем данные
ch0 = data1[1] * 256 + data1[0]
ch1 = data2[1] * 256 + data2[0]

print("Освещённость:%d люкс" %(ch0 - ch1))
```

2. ВТОРОЙ ВАРИНТ

2.1. Описание варианта

В данном варианте каждый датчик подключается к одному общему RPi как *Slave*.

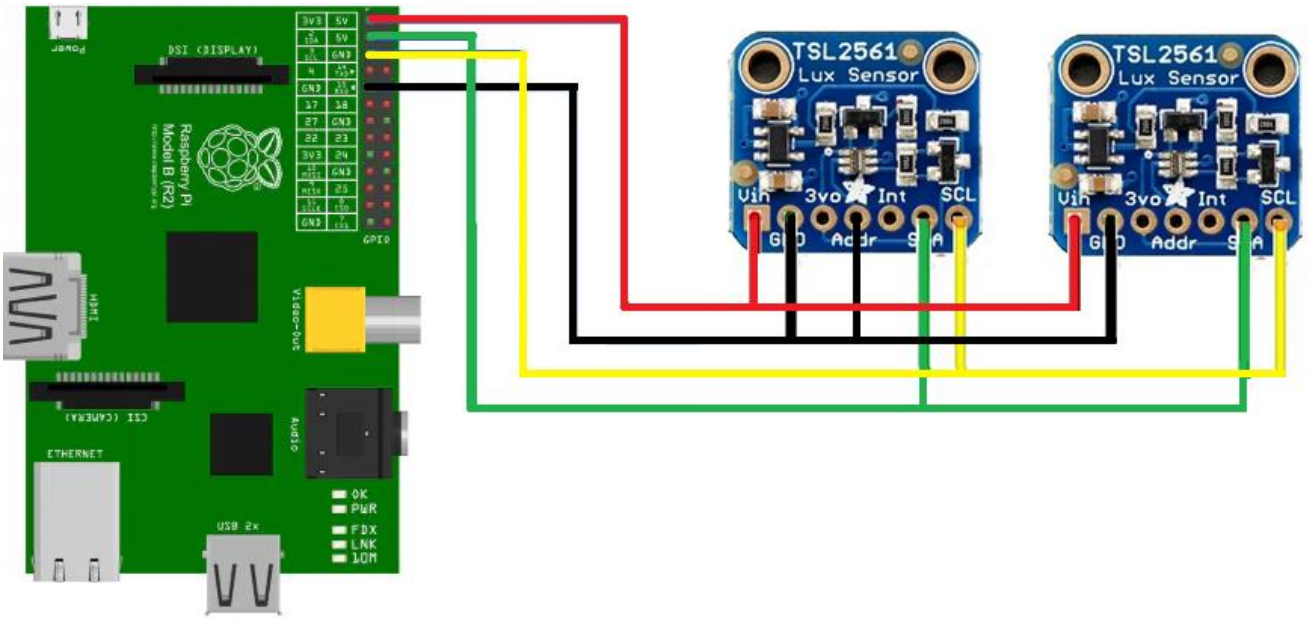
Расположение и подключение наружного датчика не изменилось по сравнению с первым вариантом, изменилось только подключение второго внутреннего датчика.



Кабель-каналы с проводами, соединяющим RPi (зелёный)
с датчиком снаружи (красный), а также со вторым внутри (оранжевый)

2.2. Физическое подключение

Во избежание коллизий адреса Slave’ов должны быть разными. Поэтому один из датчиков должен быть подключен по новой схеме. TSL2561 позволяет использовать три адреса на выбор. Для того чтобы изменить адрес с 0x39 на 0x29 необходимо лишь дополнительно соединить Addr с GND на RPi.



Датчики подключены параллельно. Подтягивающие резисторы не нужны, так как они (1.8 кОм) уже встроены в RPi.

2.3. Установка программного обеспечения

Аналогично 1.3.

2.4. Работа с датчиком

Скрипт использующий предоставляемый разработчиками драйвер представлен ниже.

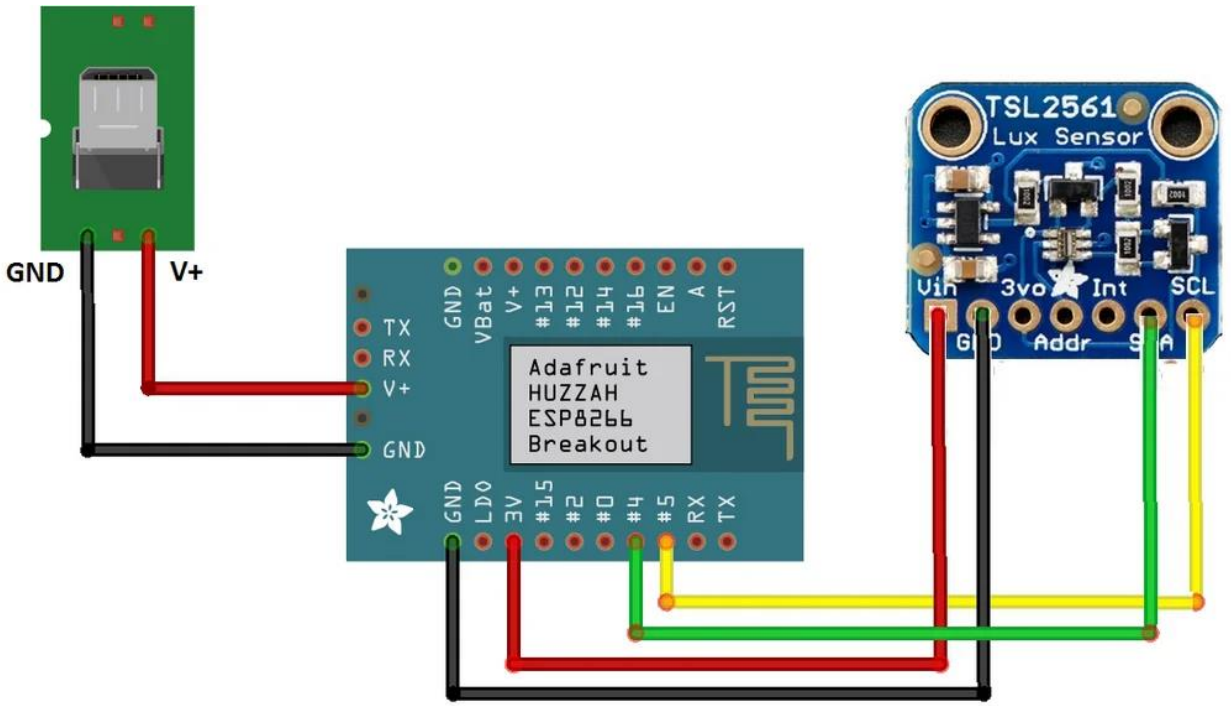
```
import time
import board
import busio
import adafruit_tsl_12561

SENSOR_1_ADDRESS = const(0x39)
SENSOR_2_ADDRESS = const(0x29)

# Создаём I2C шину
i2c = busio.I2C(board.SCL, board.SDA)

# Создаём экземпля tsl_12561 класса,
#  передаём созданную шину а также адресс перого датчика
tsl_1 = adafruit_tsl_12561.tsl_12561(i2c=i2c, address=SENSOR_1_ADDRESS)
tsl_2 = adafruit_tsl_12561.tsl_12561(i2c=i2c, address=SENSOR_2_ADDRESS)

# Включаем датчики
tsl_1.enabled = True
tsl_2.enabled = True
time.sleep(1)

# Установим усиление [ 0=1x, 1=16x ]
tsl_1.gain = 0
tsl_2.gain = 0

# Установим время интеграции (0=13.7ms, 1=101ms, 2=402ms, или 3=manual)
tsl_1.integration_time = 1
tsl_2.integration_time = 1

# Получим текущую освещённость люкс
lux_1 = tsl_1.lux
lux_2 = tsl_2.lux
```

```

# Выведем освещённость первого датчика
if lux_1 is not None:
    print("For first sensor: Lux = {}".format(lux_1))
else:
    print("For first sensor Lux value is None. "+
    "Possible sensor underrange or overrange.")

# Выведем освещённость второго датчика
if lux_2 is not None:
    print("For second sensor: Lux = {}".format(lux_2))
else:
    print("For first sensor Lux value is None. "+
    "Possible sensor underrange or overrange.")

# Выключем датчики (для экономии энергии)
tsl_1.enabled = False
tsl_2.enabled = False
```

Запуск скрипта аналогичен 1.4.

3. ТРЕТИЙ ВАРИАНТ

Если по каким-либо причинам предыдущие варианты не подходят есть также вариант с передачей данных по Wi-Fi. Для этого дополнительно понадобится Wi-Fi микроконтроллер **Adafruit HUZZAH ESP8266 breakout**. Каждый датчик подключается по схеме указной ниже.



- TSL2561 - Vin Pin ---> HUZZAH ESP8266 - 3V Pin
- TSL2561 - GND Pin ---> HUZZAH ESP8266 - GND
- TSL2561 - SDA Pin ---> HUZZAH ESP8266 - #4 Pin
- TSL2561 - SCL Pin ---> HUZZAH ESP8266 - #5 Pin

Скрипты пишутся на **MicroPython** он поддерживает работу этого чипа и имеет документацию по работе с ним. Но данный вариант гораздо сложнее в реализации чем предыдущие два.