

تعمیم



گزارش پروژه‌ی گلدان هوشمند

اعضای گروه:

روزبه پیراعیادی

آرین احدی نیا

درس:

سیستم‌های نهفته

استاد:

دکتر انصاری

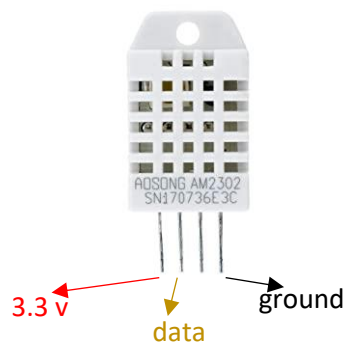
مرداد ۱۴۰۱

متصل کردن سنسور AM2302 به Raspberry pi

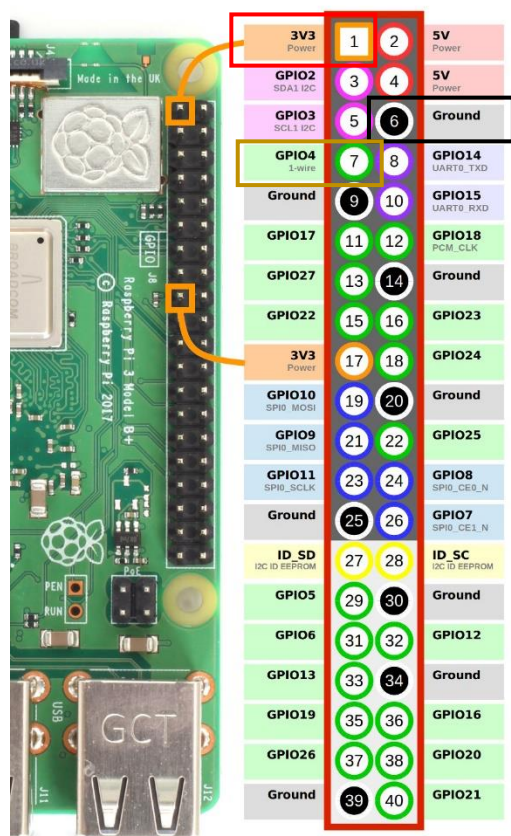
سنسور AM2302 که DHT22 نیز نامیده می‌شود یک سیگنال کم‌هزینه و دیجیتال برای اندازه‌گیری دما و رطوبت است. خروجی این سیگنال به صورت دیجیتال است. بازه‌ی این اعداد و دقت آنها به شرح زیر است.

- Humidity: 0-100%, 2-5% accuracy
- Temperature: -40 to 80°C, $\pm 0.5^{\circ}\text{C}$ accuracy

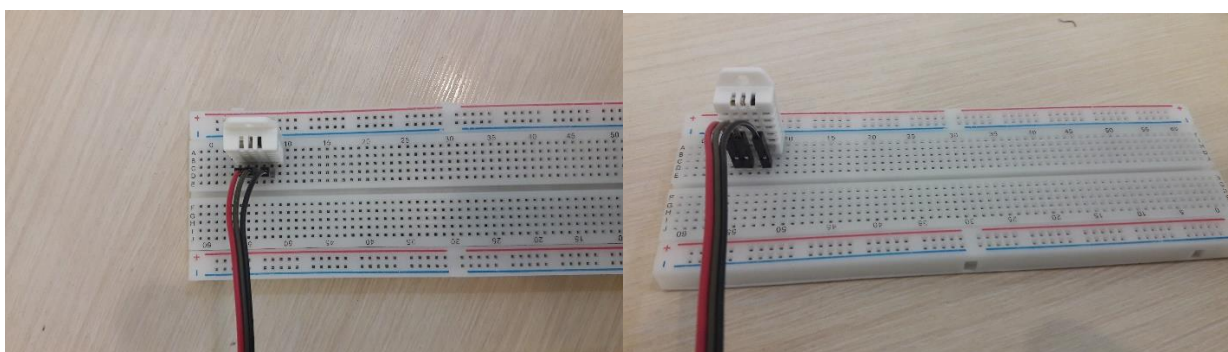
برای متصل کردن این سیگنال، سمت چپ‌ترین pin باید به 3.3 v power متصل شود، دومین pin خروجی سنسور است و سمت راست ترین pin باید به ground متصل شود.



برای اتصال Raspberry pi به سنسورها باید از GPIO pin های موجود در آن استفاده کرد. GPIO در واقع مخفف General-Purpose Input/Output و نوع آنها را در تصویر زیر مشاهده می کنید. با توجه به توضیحات بالا، pin هایی که باید از آنها استفاده کنیم مشخص شده اند. رنگ های استفاده شده متناظر با رنگ های تصویر صفحه ی قبل و همچنین رنگ سیم های به کار برده شده هستند.

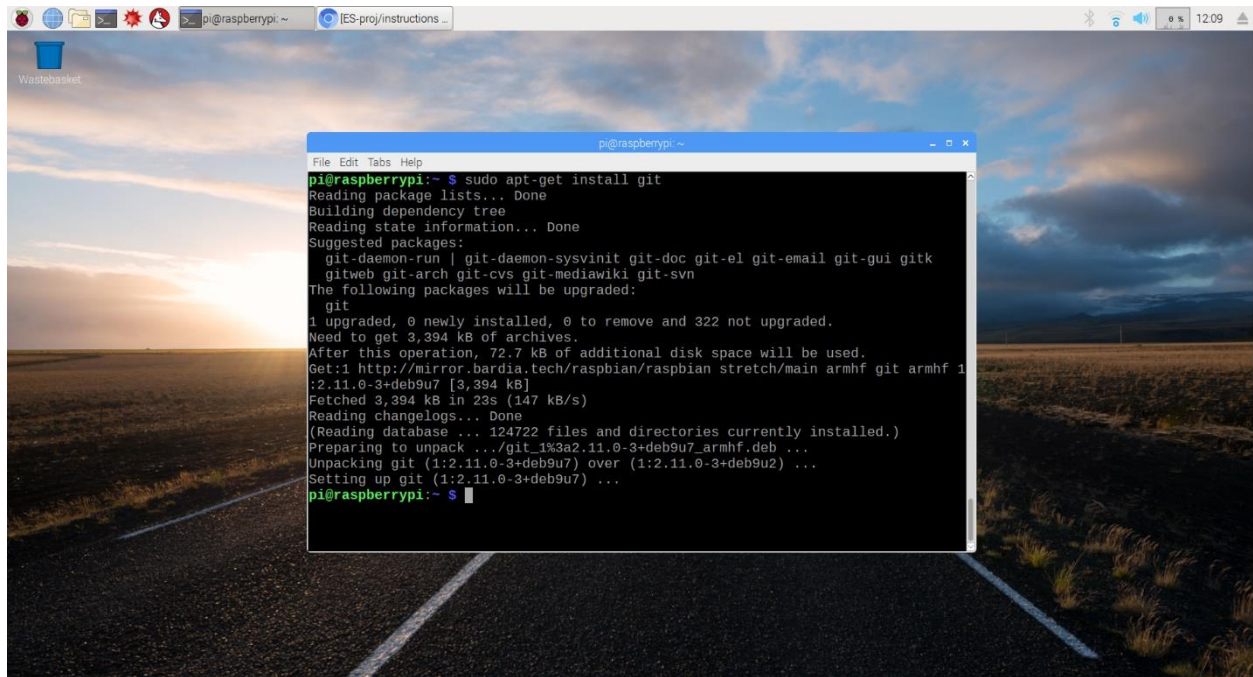


در نهایت نتیجه ی اتصال به کمک Bread Board به صورت تصاویر زیر است.



حالا برای خواندن مقادیر به صورت زیر عمل می کنیم.

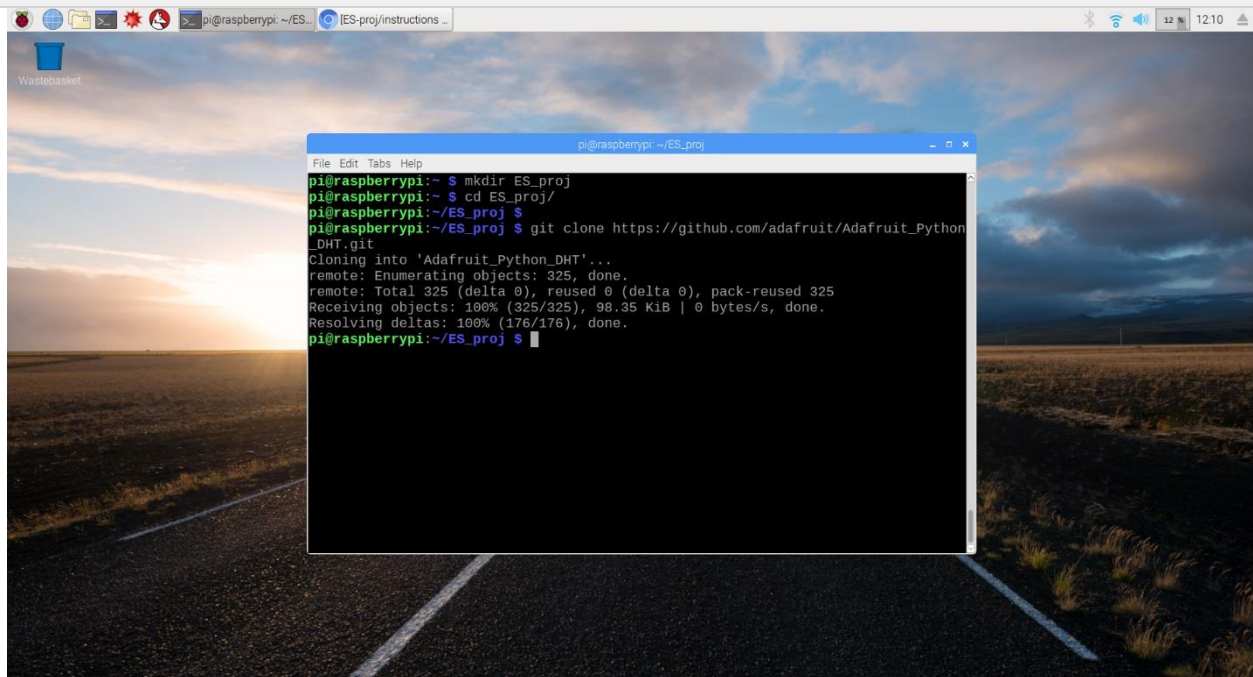
نصب کردن git



```
pi@raspberrypi:~$ sudo apt-get install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk
  gitweb git-arch git-cvs git-mediawiki git-svn
The following packages will be upgraded:
  git
1 upgraded, 0 newly installed, 0 to remove and 322 not upgraded.
Need to get 3,394 kB of archives.
After this operation, 72.7 kB of additional disk space will be used.
Get:1 http://mirror.bardia.tech/raspbian/raspbian stretch/main armhf git armhf 1:2.11.0-3+deb9u7 [3,394 kB]
Fetched 3,394 kB in 23s (147 kB/s)
Reading changelogs... Done
(Reading database ... 124722 files and directories currently installed.)
Preparing to unpack .../git_1%3a2.11.0-3+deb9u7_armhf.deb ...
Unpacking git (1:2.11.0-3+deb9u7) over (1:2.11.0-3+deb9u2) ...
Setting up git (1:2.11.0-3+deb9u7) ...
pi@raspberrypi:~$
```

ساختن یک پوشه با نام ES_proj و clone کردن پروژه adafruit با کمک دستور زیر

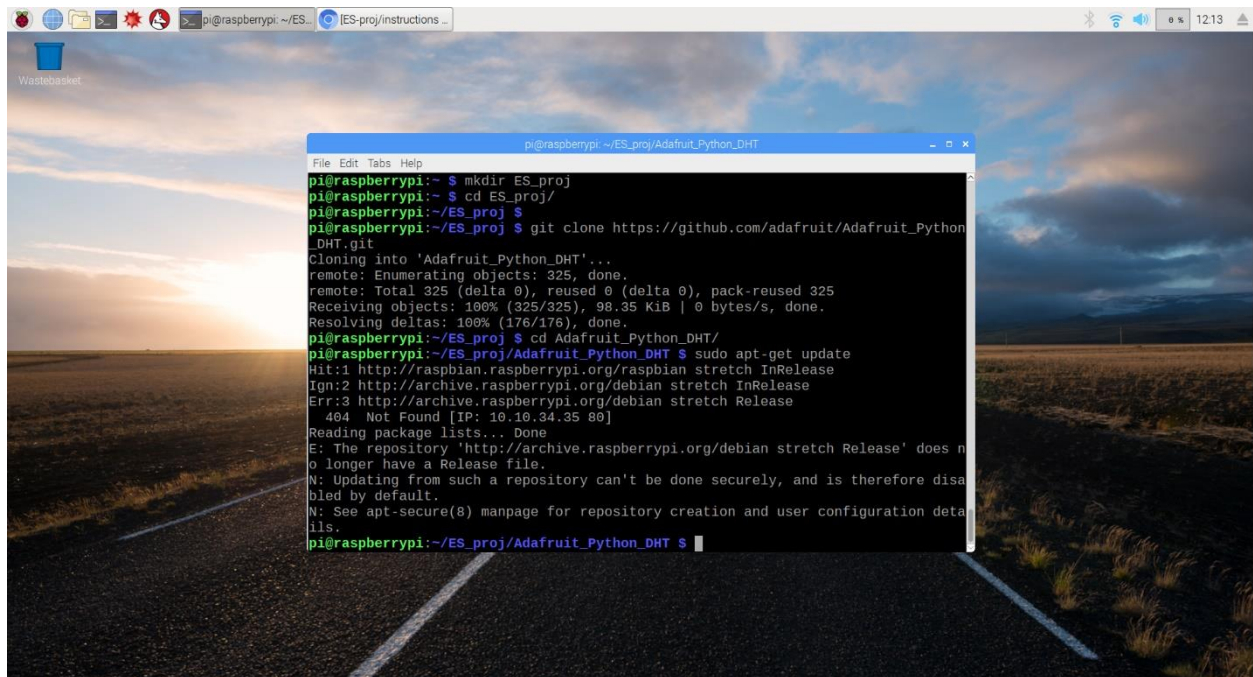
```
git clone https://github.com/adafruit/Adafruit_Python_DHT.git
```



```
pi@raspberrypi:~$ mkdir ES_proj
pi@raspberrypi:~$ cd ES_proj/
pi@raspberrypi:~/ES_proj$ git clone https://github.com/adafruit/Adafruit_Python_DHT.git
Cloning into 'Adafruit_Python_DHT'...
remote: Enumerating objects: 325, done.
remote: Total 325 (delta 0), reused 0 (delta 0), pack-reused 325
Receiving objects: 100% (325/325), 98.35 KiB | 0 bytes/s, done.
Resolving deltas: 100% (176/176), done.
pi@raspberrypi:~/ES_proj$
```



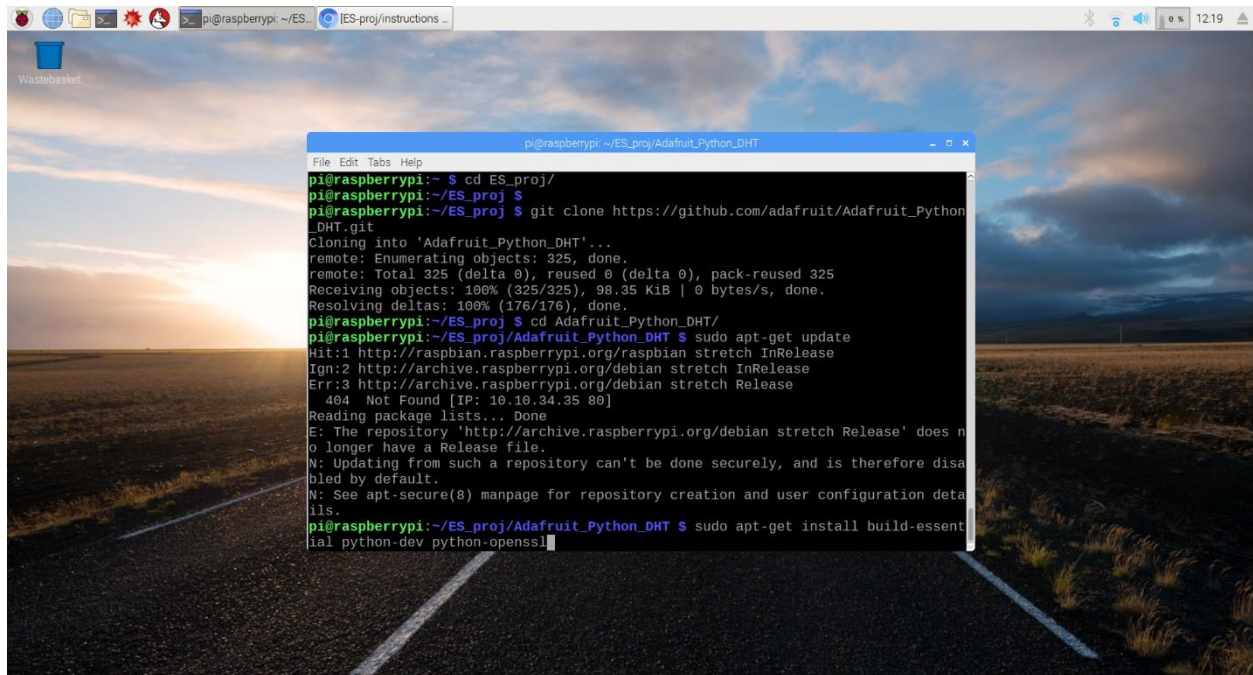
```
sudo apt-get update
```



The screenshot shows a Raspberry Pi desktop with a terminal window open. The terminal displays the following commands and output:

```
pi@raspberrypi: ~/ES_proj
pi@raspberrypi:~/ES_proj$ mkdir ES_proj
pi@raspberrypi:~/ES_proj$ cd ES_proj/
pi@raspberrypi:~/ES_proj$ git clone https://github.com/adafruit/Adafruit_Python_DHT.git
Cloning into 'Adafruit_Python_DHT'...
remote: Enumerating objects: 325, done.
remote: Total 325 (delta 0), reused 0 (delta 0), pack-reused 325
Receiving objects: 100% (325/325), 98.35 KiB | 0 bytes/s, done.
Resolving deltas: 100% (176/176), done.
pi@raspberrypi:~/ES_proj$ cd Adafruit_Python_DHT/
pi@raspberrypi:~/ES_proj/Adafruit_Python_DHT$ sudo apt-get update
Hit:1 http://raspbian.raspberrypi.org/raspbian stretch InRelease
Ign:2 http://archive.raspberrypi.org/debian stretch InRelease
Err:3 http://archive.raspberrypi.org/debian stretch Release
404 Not Found [IP: 10.10.34.35 80]
Reading package lists... Done
E: The repository 'http://archive.raspberrypi.org/debian stretch Release' does not
  longer have a Release file.
N: Updating from such a repository can't be done securely, and is therefore disa
  bled by default.
N: See apt-secure(8) manpage for repository creation and user configuration deta
  ils.
pi@raspberrypi:~/ES_proj/Adafruit_Python_DHT$
```

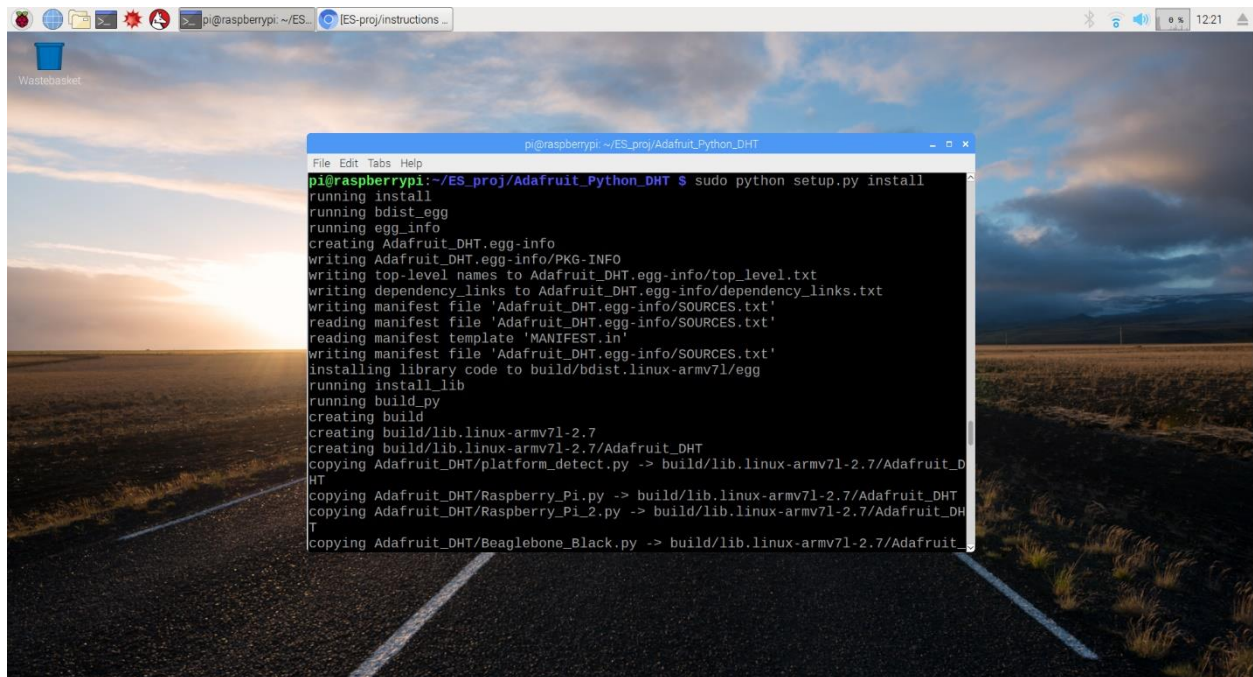
```
sudo apt-get install build-essential python-dev python-openssl
```



The screenshot shows a Raspberry Pi desktop with a terminal window open. The terminal displays the following commands and output:

```
pi@raspberrypi: ~/ES_proj
pi@raspberrypi:~/ES_proj$ cd ES_proj/
pi@raspberrypi:~/ES_proj$ git clone https://github.com/adafruit/Adafruit_Python_DHT.git
Cloning into 'Adafruit_Python_DHT'...
remote: Enumerating objects: 325, done.
remote: Total 325 (delta 0), reused 0 (delta 0), pack-reused 325
Receiving objects: 100% (325/325), 98.35 KiB | 0 bytes/s, done.
Resolving deltas: 100% (176/176), done.
pi@raspberrypi:~/ES_proj$ cd Adafruit_Python_DHT/
pi@raspberrypi:~/ES_proj/Adafruit_Python_DHT$ sudo apt-get update
Hit:1 http://raspbian.raspberrypi.org/raspbian stretch InRelease
Ign:2 http://archive.raspberrypi.org/debian stretch InRelease
Err:3 http://archive.raspberrypi.org/debian stretch Release
404 Not Found [IP: 10.10.34.35 80]
Reading package lists... Done
E: The repository 'http://archive.raspberrypi.org/debian stretch Release' does not
  longer have a Release file.
N: Updating from such a repository can't be done securely, and is therefore disa
  bled by default.
N: See apt-secure(8) manpage for repository creation and user configuration deta
  ils.
pi@raspberrypi:~/ES_proj/Adafruit_Python_DHT$ sudo apt-get install build-essent
ial python-dev python-openssl
```

```
sudo python setup.py install
```



```
pi@raspberrypi: ~/ES_proj/Adafruit_Python_DHT
File Edit Tabs Help
pi@raspberrypi:~/ES_proj/Adafruit_Python_DHT $ sudo python setup.py install
running install
running bdist_egg
running egg_info
creating Adafruit_DHT.egg-info
writing Adafruit_DHT.egg-info/PKG-INFO
writing top-level names to Adafruit_DHT.egg-info/top_level.txt
writing dependency links to Adafruit_DHT.egg-info/dependency_links.txt
writing manifest file 'Adafruit_DHT.egg-info/SOURCES.txt'
reading manifest file 'Adafruit_DHT.egg-info/SOURCES.txt'
reading manifest template 'MANIFEST.in'
writing manifest file 'Adafruit_DHT.egg-info/SOURCES.txt'
installing library code to build/bdist.linux-armv7l/egg
running install_lib
running build_py
creating build
creating build/lib.linux-armv7l-2.7
creating build/lib.linux-armv7l-2.7/Adafruit_DHT
copying Adafruit_DHT/platform_detect.py -> build/lib.linux-armv7l-2.7/Adafruit_DHT
copying Adafruit_DHT/Raspberry_Pi.py -> build/lib.linux-armv7l-2.7/Adafruit_DHT
copying Adafruit_DHT/Raspberry_Pi_2.py -> build/lib.linux-armv7l-2.7/Adafruit_DHT
copying Adafruit_DHT/Beaglebone_Black.py -> build/lib.linux-armv7l-2.7/Adafruit_DHT
```

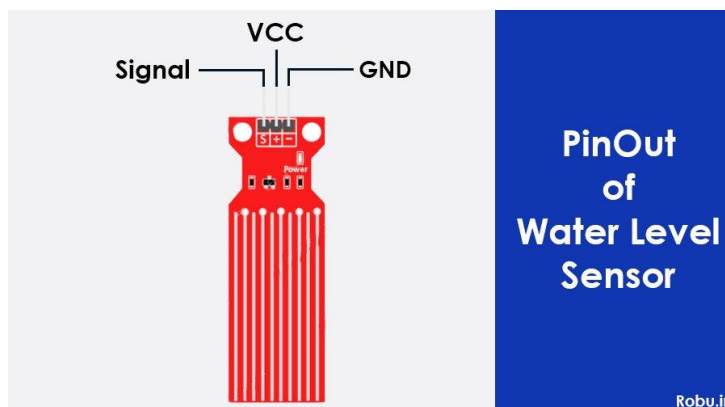
تمامی دستورات این بخش به صورت تجمیعی در ادامه آمده‌اند.

```
git clone https://github.com/adafruit/Adafruit_Python_DHT.git
cd Adafruit_Python_DHT
sudo apt-get update
sudo apt-get install build-essential python-dev
sudo python setup.py install
```

بعدتر کد پایتونی که با کمک کتابخانه‌ی AdaFruit خروجی سنسور را می‌گیرد را شرح خواهیم داد.

متصل کردن سنسور سطح آب به Raspberry pi

ابتدا به نوع pin های سنسور سطح آب که در تصویر زیر نشان داده شده‌اند توجه کنید.

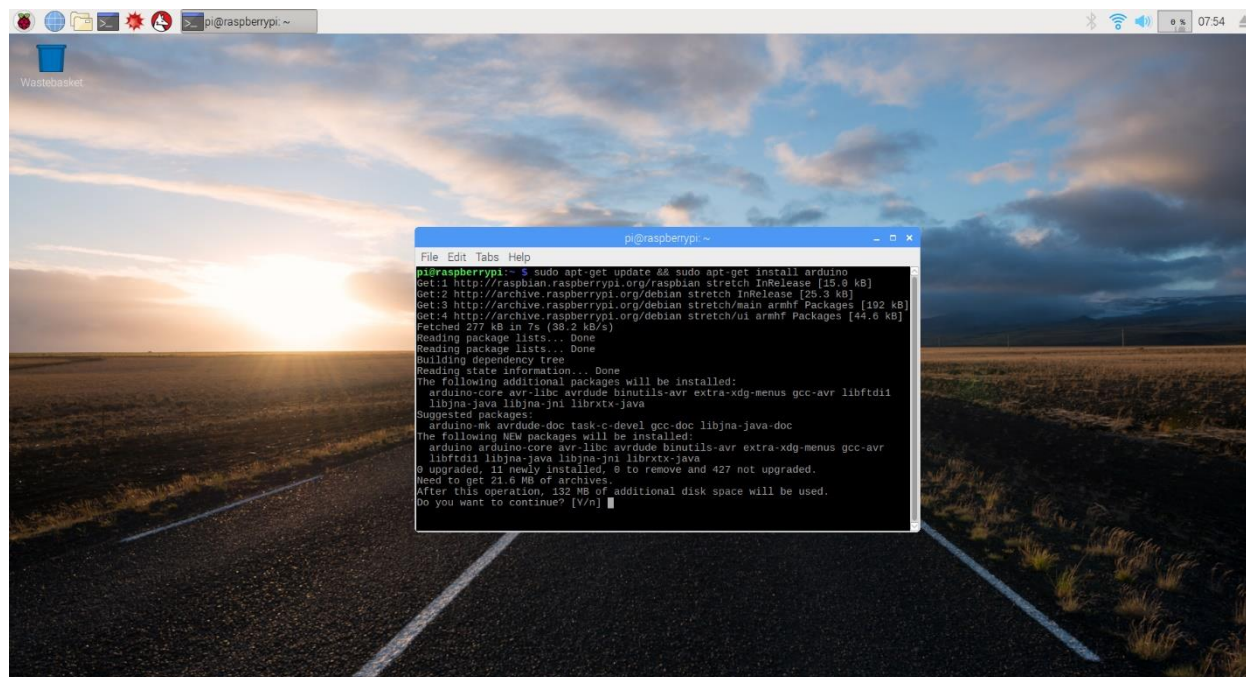


اما موضوعی که وجود دارد این است که خروجی این سنسور به صورت آنالوگ است و برای متصل کردن این سنسور به Raspberry pi باید به نوعی خروجی آن را به یک خروجی دیجیتال تبدیل کنیم. در این جا از Arduino uno برای این کار استفاده می‌کنیم. بنابراین اتصالات را باید به شکل زیر انجام دهیم. همان‌طور که می‌بینید سیم زرد به پین آنالوگ روی arduino، سیم نارنجی به 5v و سیم قرمز به GND متصل شده است.



برای این که بتوانیم از خروجی Arduino روی raspberry pi استفاده کنیم، ابتدا لازم است تا ide مربوط به کار کردن با arduino را روی سیستم عامل raspbian نصب کنیم.

```
sudo apt-get update && sudo apt-get install arduino
```



سپس با نوشتن کد زیر درون این محیط می‌توانیم، سپس compile کردن و آپلود کردن آن روی arduino می‌توانیم خروجی سنسور را به صورت نسبی مشاهده کنیم.


```
File Edit Sketch Tools Help
sketch_jul14a s
int resval = 0;
int respin = A0;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  resval = analogRead(respin);

  if(resval <= 400){
    Serial.println("water level: NO Water");
    Serial.println(resval);
  }
  else if (resval > 300 && resval <= 500){
    Serial.println("water level: Low");
    Serial.println(resval);
  }
  else if (resval > 500 && resval <= 650){
    Serial.println("water level: Medium");
    Serial.println(resval);
  }
  else if (resval > 650){
    Serial.println("water level: High");
    Serial.println(resval);
  }
  delay(1000);
}

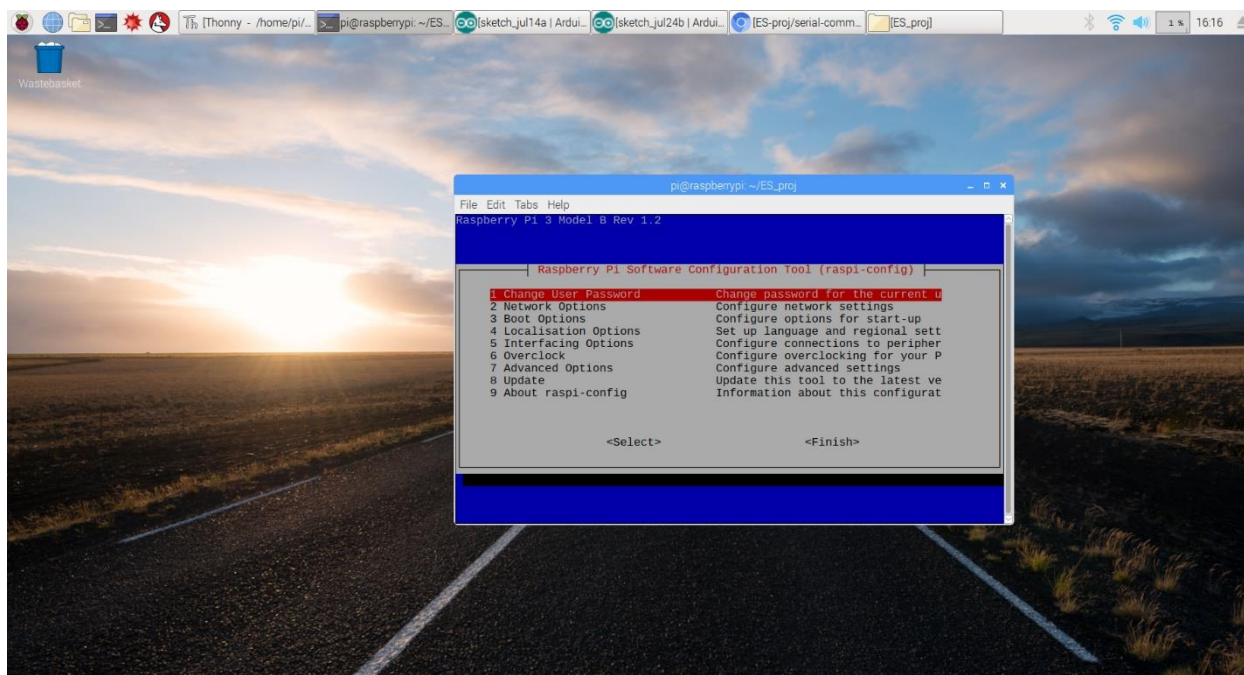
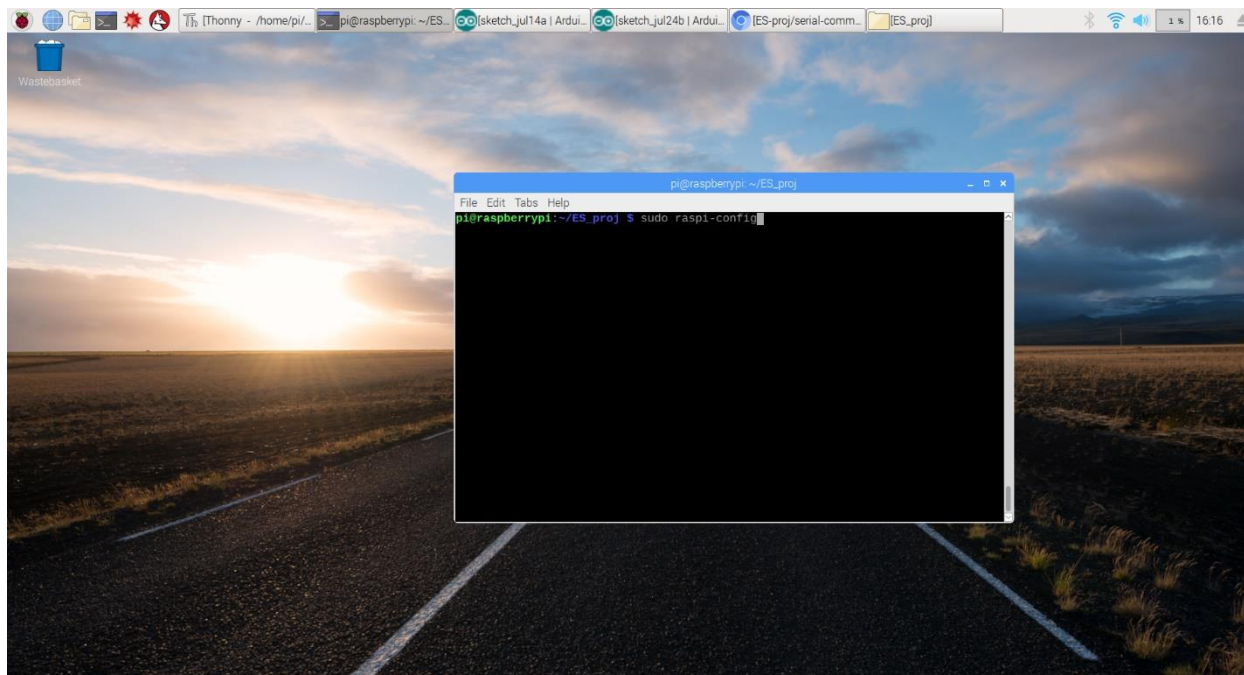
The library "HttpClient-2.2.0" cannot be used. Library names must contain only basic letters and numbers (ASCII only and no spaces, and it cannot start with a number)

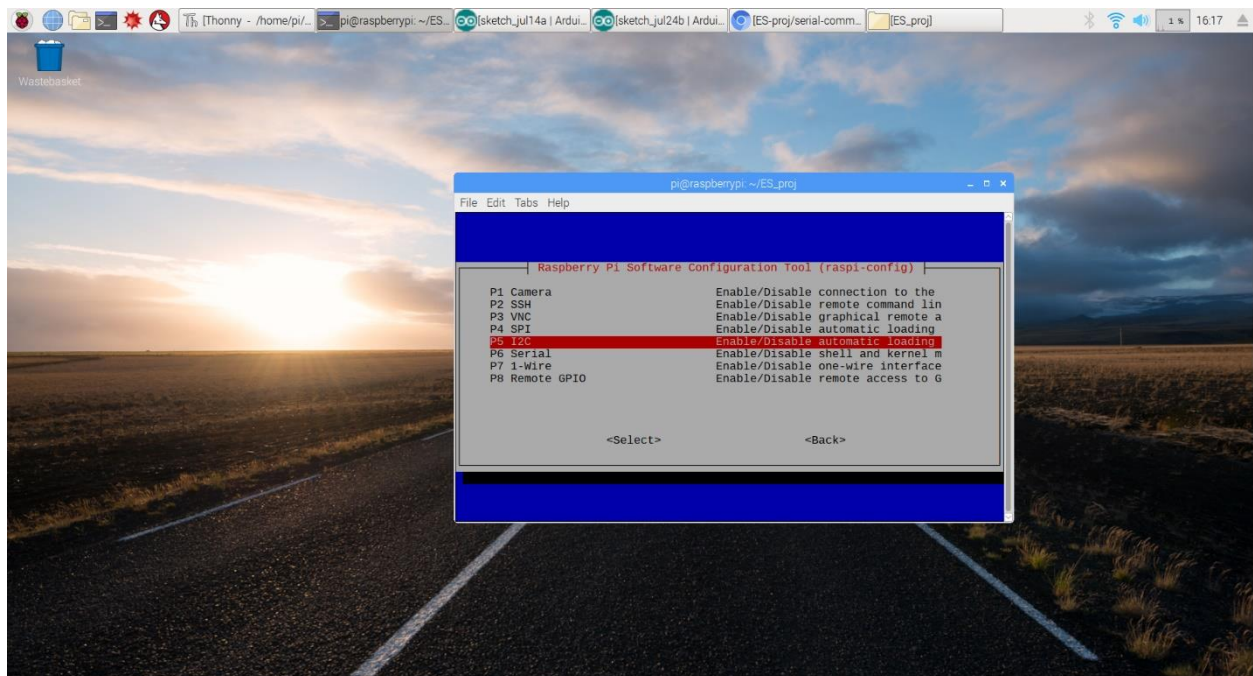
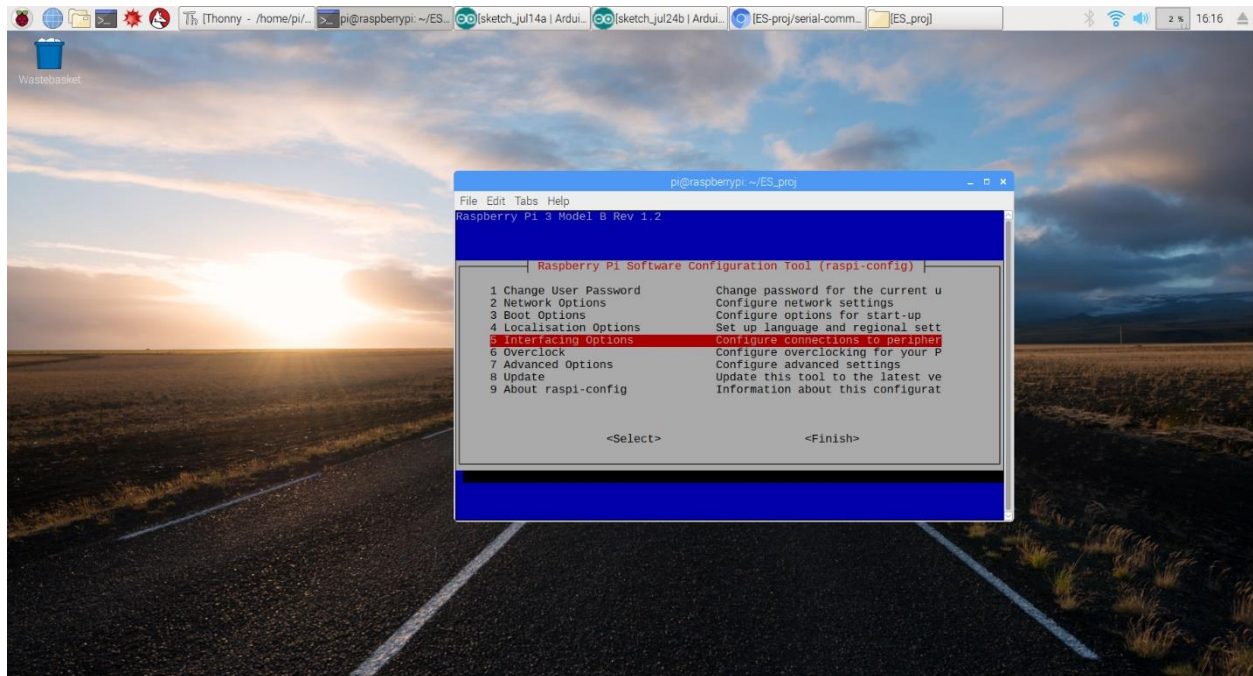
Arduino Uno on /dev/ttyACM0
```

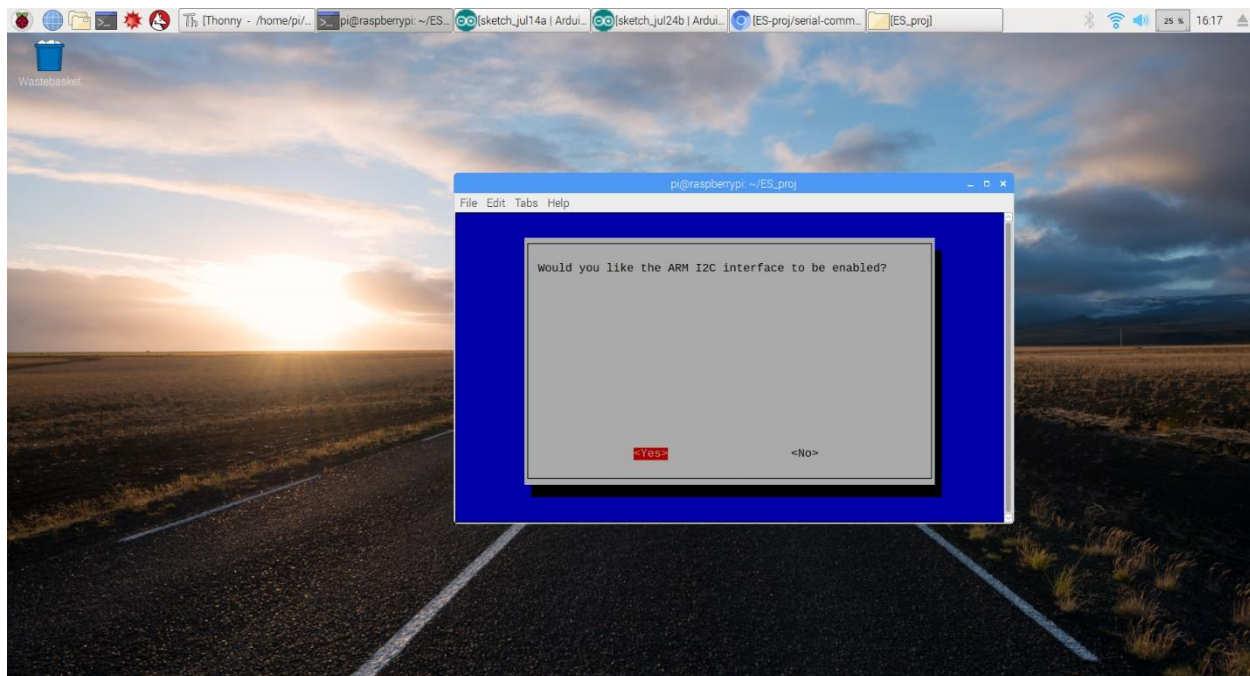
دو نکته در کد بالا وجود دارد. یکی این است که **respin** که خواندن از روی آن انجام می‌شود باید دقیقاً همان **pin** باشد که سنسور را بدان متصل کرده‌ایم. دوم این که **9600** موجود در خط **Serial.begin(9600)** baud rate است. این مقدار به این معناست که اطلاعات با نرخ **9600** بیت در ثانیه منتقل می‌شوند.

اما داشتن اطلاعات روی arduino کافی نیست و باید آن را به raspberry pi منتقل کنیم. برای این کار در ابتدا به صورت زیر I2C serial communication protocol را فعال می‌کنیم.

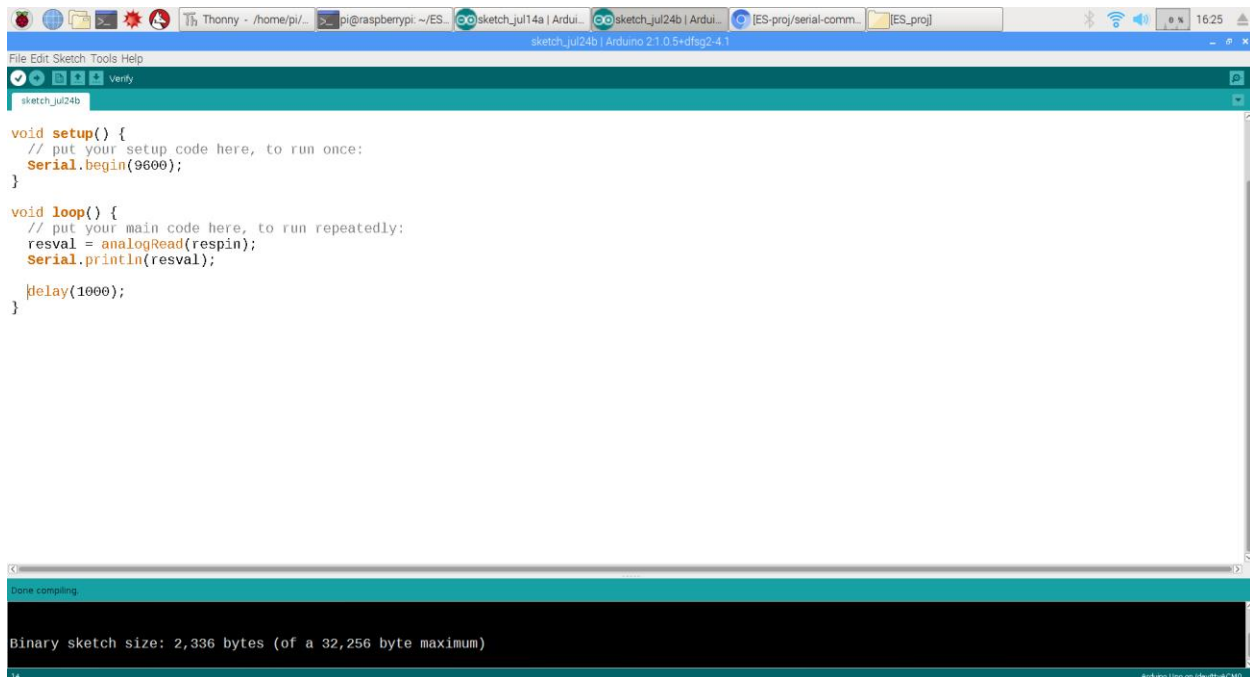
```
sudo raspi-config
```







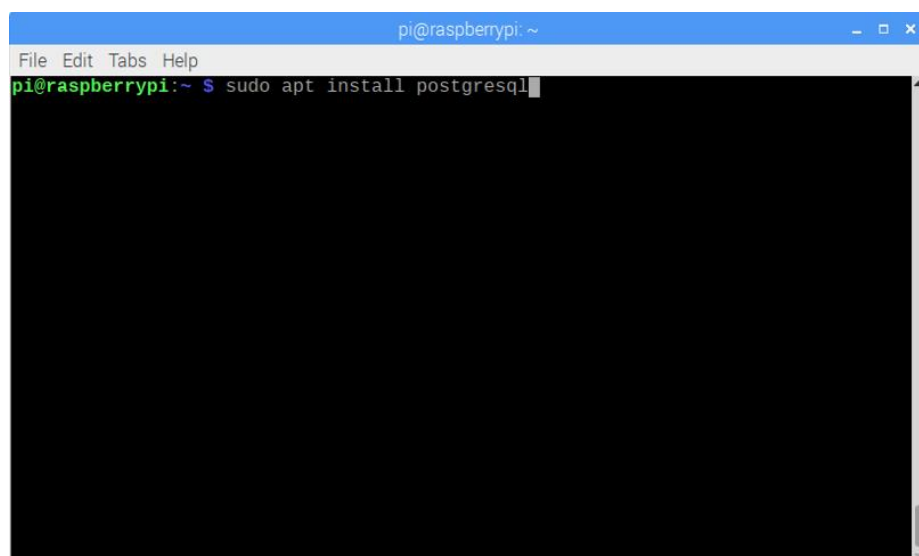
حالا که می‌خواهیم اطلاعات را به Raspberry pi منتقل کنیم دیگر احتیاج به انجام پردازش اضافه روی داده‌ها نیست و می‌توانیم با استفاده از کد زیر، آنها را به صورت خام ارسال کنیم.



نصب کردن postgresql

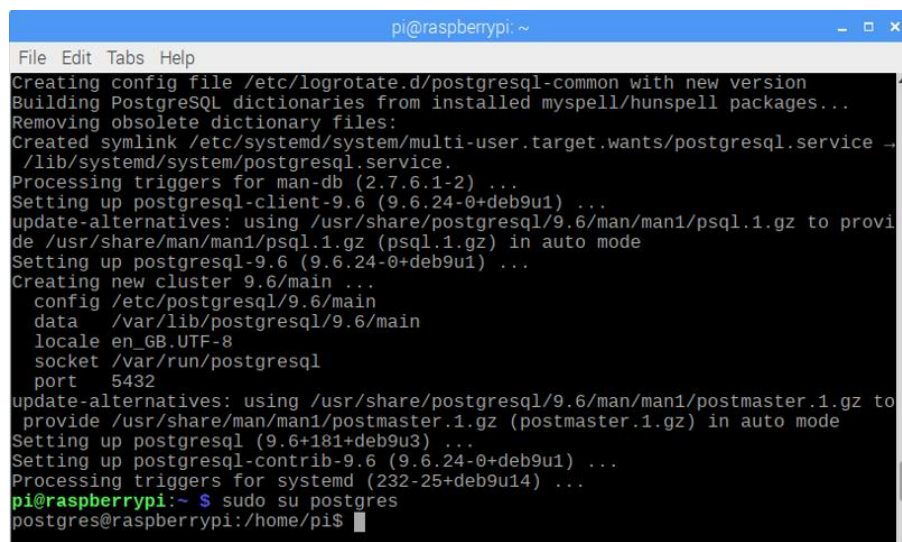
برخی اوقات به دلایل مختلف نظیر قطع بودن اتصال شبکه و یا مشکل در سرور ممکن است ارسال اطلاعات موفقیت آمیز نباشد. بنابراین برای این که اطلاعات خوانده شده از بین نروند، لازم است تا اطلاعاتی که موفق به ارسال آنها نشده ایم را در یک پایگاه داده ذخیره کنیم. برای نصب کردن postgresql به صورت زیر عمل می کنیم.

```
sudo apt update
sudo apt upgrade
sudo apt install postgresql
```

A terminal window titled 'pi@raspberrypi: ~' with a menu bar (File, Edit, Tabs, Help). The command 'pi@raspberrypi:~ \$ sudo apt install postgresql' is entered and the cursor is at the end of the line.

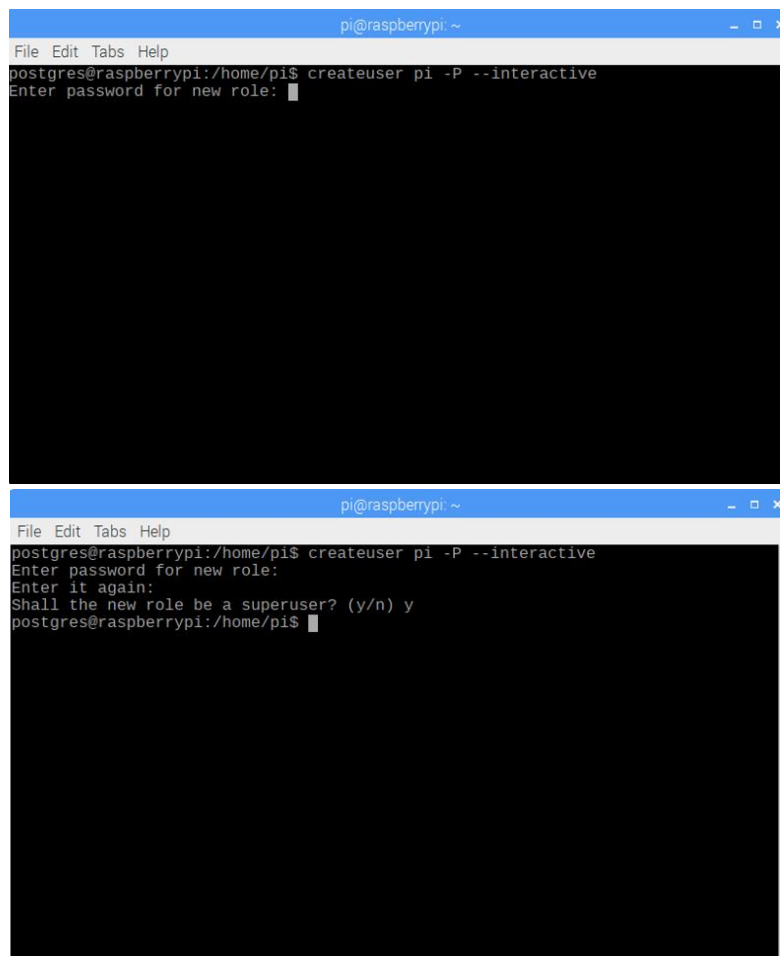
```
pi@raspberrypi:~ $ sudo apt install postgresql
```

```
sudo su postgres
```

A terminal window titled 'pi@raspberrypi: ~' showing the output of the 'sudo apt install postgresql' command. The output includes messages about creating config files, building dictionaries, removing obsolete files, creating symlinks, processing triggers, setting up client and server components, and creating a new cluster. The prompt changes to 'postgres@raspberrypi:/home/pi\$' at the bottom.

```
pi@raspberrypi:~ $ sudo apt install postgresql
Creating config file /etc/logrotate.d/postgresql-common with new version
Building PostgreSQL dictionaries from installed myspell/hunspell packages...
Removing obsolete dictionary files:
Created symlink /etc/systemd/system/multi-user.target.wants/postgresql.service →
/lib/systemd/system/postgresql.service.
Processing triggers for man-db (2.7.6.1-2) ...
Setting up postgresql-client-9.6 (9.6.24-0+deb9u1) ...
update-alternatives: using /usr/share/postgresql/9.6/man/man1/psql.1.gz to provi
de /usr/share/man/man1/psql.1.gz (psql.1.gz) in auto mode
Setting up postgresql-9.6 (9.6.24-0+deb9u1) ...
Creating new cluster 9.6/main ...
   config /etc/postgresql/9.6/main
   data   /var/lib/postgresql/9.6/main
   locale en_GB.UTF-8
   socket /var/run/postgresql
   port   5432
update-alternatives: using /usr/share/postgresql/9.6/man/man1/postmaster.1.gz to
 provide /usr/share/man/man1/postmaster.1.gz (postmaster.1.gz) in auto mode
Setting up postgresql (9.6+181+deb9u3) ...
Setting up postgresql-contrib-9.6 (9.6.24-0+deb9u1) ...
Processing triggers for systemd (232-25+deb9u14) ...
pi@raspberrypi:~ $ sudo su postgres
postgres@raspberrypi:/home/pi$
```

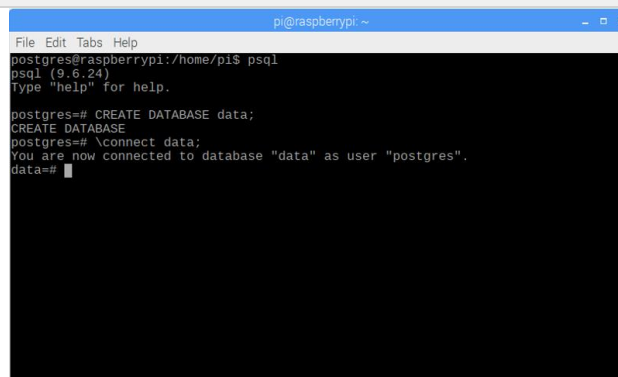
```
createuser pi -P --interactive
```



```
pi@raspberrypi: ~  
File Edit Tabs Help  
postgres@raspberrypi:/home/pi$ createuser pi -P --interactive  
Enter password for new role:   
  
postgres@raspberrypi:/home/pi$ createuser pi -P --interactive  
Enter password for new role:  
Enter it again:  
Shall the new role be a superuser? (y/n) y  
postgres@raspberrypi:/home/pi$
```

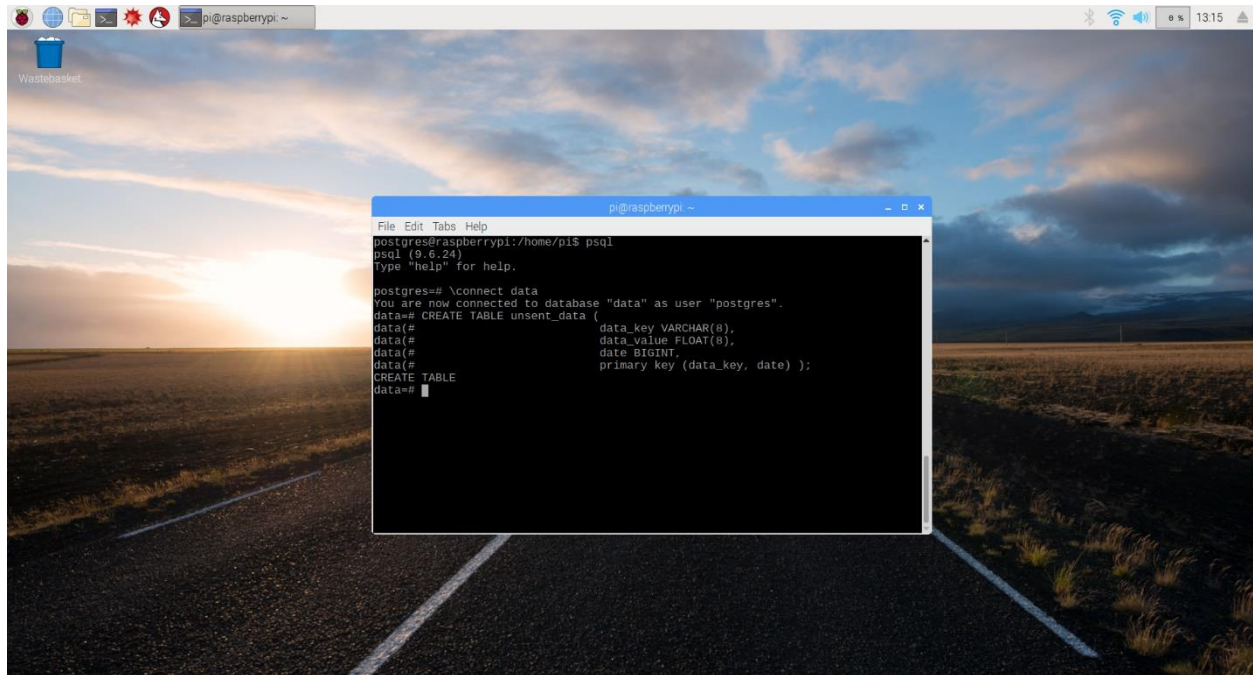
حالا می‌توانیم یک دیتابیس ایجاد کنیم و به آن متصل شویم.

```
psql  
CREATE DATABASE [NAME]  
\connect [NAME]
```



```
pi@raspberrypi: ~  
File Edit Tabs Help  
postgres@raspberrypi:/home/pi$ psql  
psql (9.6.24)  
Type "help" for help.  
  
postgres=# CREATE DATABASE data;  
CREATE DATABASE  
postgres=# \connect data;  
You are now connected to database "data" as user "postgres".  
data=#
```

در انتها یک جدول ایجاد می‌کنیم.



در نهایت لازم است تا کتابخانه‌ی `psycopg2` را نصب کنیم تا بتوانیم از طریق کد پایتون به دیتابیس متصل شویم.

```
sudo apt-get install postgresql  
sudo apt-get install python-psycopg2  
sudo apt-get install libpq-dev
```

```
pi@raspberrypi: ~/ES_proj/code  
File Edit Tabs Help  
  
pi@raspberrypi:~/ES_proj/code $ sudo apt-get install postgresql  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
postgresql is already the newest version (9.6+181+deb9u3).  
The following packages were automatically installed and are no longer required:  
  coinor-libipopt1v5 libmumps-seq-4.10.0 libraw15 lxkeymap python-cairo  
  python-gobject python-gobject-2 python-gtk2 python-xklavier realpath  
Use 'sudo apt autoremove' to remove them.  
0 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.  
pi@raspberrypi:~/ES_proj/code $ sudo apt-get install python-psycopg2  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following packages were automatically installed and are no longer required:  
  coinor-libipopt1v5 libmumps-seq-4.10.0 libraw15 lxkeymap python-cairo  
  python-gobject python-gobject-2 python-gtk2 python-xklavier realpath  
Use 'sudo apt autoremove' to remove them.  
The following additional packages will be installed:  
  python-egenix-mxdatetime python-egenix-mxtools  
Suggested packages:  
  python-egenix-mxdatetime-dbg python-egenix-mxdatetime-doc  
  python-egenix-mxtools-dbg python-egenix-mxtools-doc python-psycopg2-doc
```

```
pi@raspberrypi: ~/ES_proj/code
File Edit Tabs Help
pi@raspberrypi:~/ES_proj/code $ sudo apt-get install libpq-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  coinor-libipopt1v5 libmumps-seq-4.10.0 libraw15 lxkeymap python-cairo
  python-gobject python-gobject-2 python-gtk2 python-xklavier realpath
Use 'sudo apt autoremove' to remove them.
Suggested packages:
  postgresql-doc-9.6
The following NEW packages will be installed:
  libpq-dev
0 upgraded, 1 newly installed, 0 to remove and 2 not upgraded.
Need to get 198 kB of archives.
After this operation, 819 kB of additional disk space will be used.
Get:1 http://mirror.bardiac.tech/raspbian/raspbian stretch/main armhf libpq-dev a
rmhf 9.6.24-0+deb9u1 [198 kB]
Fetched 198 kB in 1s (165 kB/s)
Selecting previously unselected package libpq-dev.
(Reading database ... 136242 files and directories currently installed.)
Preparing to unpack .../libpq-dev_9.6.24-0+deb9u1_armhf.deb ...
Unpacking libpq-dev (9.6.24-0+deb9u1) ...
Setting up libpq-dev (9.6.24-0+deb9u1) ...
Processing triggers for man-db (2.7.6.1-2) ...
```


تشریح کدهای پایتون نوشته شده

send_data.py

ابتدا به توضیح کد send_data.py می پردازیم. این کد وظیفه‌ی خواندن اطلاعات از سنسورها و فرستادن آنها به سرور را بر عهده دارد. همچنین در صورت موفقیت آمیز نبودن ارتباط با سرور انتظار می رود اطلاعات خوانده شده در پایگاه داده ذخیره شوند.

در بخش اول کتابخانه‌های لازم را import می کنیم. سپس ورودی ها را از ترمینال دریافت می کنیم. انتظار داریم ورودی اول یکی از سه عدد 11، 12 و 2302 باشد که ورودی مربوط به سنسور ما 2302 است. ورودی دوم نیز شماره‌ی pin را مشخص می کند که اگر به بخش اول مراجعه کنید، خواهید دید که سنسور دما و رطوبت به pin شماره‌ی 4 متصل شده است.

```
import sys
import requests
import Adafruit_DHT
import time
import serial
import string
import psycpg2

sensor_args = { '11': Adafruit_DHT.DHT11,
                 '22': Adafruit_DHT.DHT22,
                 '2302': Adafruit_DHT.AM2302 }
if len(sys.argv) == 3 and sys.argv[1] in sensor_args:
    sensor = sensor_args[sys.argv[1]]
    pin = sys.argv[2]
else:
    print('Usage: sudo ./Adafruit_DHT.py [11|22|2302] <GPIO pin number>')
    print('Example: sudo ./Adafruit_DHT.py 2302 4 - Read from an AM2302 connected to GPIO pin #4')
    sys.exit(1)
```

در بخش بعد یک تابع برای اتصال به پایگاه داده و یک تابع با نام insert_into_database نوشته شده است. این تابع به پایگاه داده unsent_data یک سطر اضافه می کند. data_key یکی از سه مقدار Temp، Hum و Lev است که به ترتیب دما، رطوبت و سطح آب را نشان می دهد.

```
def connect_to_database():
    connection = None
    try:
        connection = psycpg2.connect(user = 'pi',
                                     password = '1234',
                                     host = 'localhost',
                                     port = '5432',
                                     database = 'data')

        cur = connection.cursor()

    except (Exception, psycpg2.DatabaseError) as error:
        print(error)

    return connection, cur

def insert_into_database(data_key, data_value, date):
    sql = """INSERT INTO unsent_data
              VALUES(%s, %s, %s)"""

    connection, cur = connect_to_database()

    if connection is not None:
        cur.execute(sql, (data_key, data_value, date))
        connection.commit()
        cur.close()
        connection.close()
```

تابع check_status_code بررسی می کند که status_code برگردانده شده توسط سرور، successful بوده است یا خیر. تابع send_or_save_data نیز به کمک دو تابع check_satus_code و insert_into_database یک داده به صورت

dictionary را دریافت می‌کند و یا آن را ارسال می‌کند یا در پایگاه داده ذخیره می‌کند. چند خط بعد هم برای خواندن device_serial و private_key هستند. device_serial همان شماره سریالی است که به کیت اختصاص داده شده است. private_key نیز کلیدی است که سرور به کمک آن صحت اطلاعات ارسالی را بررسی می‌کند تا در صورتی که اطلاعات غلطی توسط فرستنده‌های دیگر به سرور ارسال شود، این اطلاعات در نظر گرفته نشوند.

```
def check_status_code(status_code):
    if status_code == 200:
        return True
    else:
        return False

def send_or_save_data(url, sensor_data):
    print(sensor_data)
    result = requests.post(url, data = sensor_data)
    if not check_status_code(result.status_code):
        print('Couldnt send Data. Data was saved in database')
        insert_into_database(sensor_data['dataKey'], sensor_data['dataValue'], sensor_data['timeInstant'])

f1 = open('./info/device_serial', 'r')
device_serial = f1.read()
f1.close()

f2 = open('./info/private_key', 'r')
private_key = f2.read()
f2.close()
```

در نهایت به بخش اصلی می‌رسیم. ابتدا به کمک کتابخانه‌ی Adafruit اقدام به خواندن دما و رطوبت می‌کنیم. برای خواندن سطح آب هم کافیست به خواندن به صورت سریال از port usb با baudrate 9600 پردازیم. سپس با کمک send_or_save_data یا تمام داده‌ها را می‌فرستیم یا آنها را در پایگاه داده ذخیره می‌کنیم. همان طور که می‌بینید این کار هر ۱۰ ثانیه انجام می‌شود. هر چند این مقدار کوچکتر از مقداری است که در دنیای حقیقی لازم است، اما برای آزمایش ساده‌تر سیستم قرار داده شده است.

```
ser=serial.Serial('/dev/ttyACM0', 9600)
url = 'http://94.101.178.12/api/post'
try:
    while True:
        humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
        if humidity == None or temperature == None:
            print('Data Read Failed')
            continue

        print('Data Read Finish')

        water_level = int(ser.readline().strip())

        temp_data = {
            'deviceSerial': int(device_serial),
            'privateKey' : private_key,
            'timeInstant' : int(time.time() * 1000),
            'dataKey': 'Temp',
            'dataValue' : temperature}

        hum_data = {
            'deviceSerial': int(device_serial),
            'privateKey' : private_key,
            'timeInstant' : int(time.time() * 1000),
            'dataKey': 'Hum',
            'dataValue' : humidity}

        lev_data = {
            'deviceSerial': int(device_serial),
            'privateKey' : private_key,
            'timeInstant' : int(time.time() * 1000),
            'dataKey': 'Lev',
            'dataValue' : water_level}

        send_or_save_data(url, temp_data)
        send_or_save_data(url, hum_data)
        send_or_save_data(url, lev_data)

        time.sleep(10)
except KeyboardInterrupt:
    pass
```

send_unsent_data.py

همان طور که در بخش قبل شرح داده شد، ممکن است برخی داده‌ها به هر دلیل ارسال نشده باشند و در پایگاه داده ذخیره شده باشند. این کد وظیفه‌ی این را دارد که به صورت دوره ای ارتباط با سرور بررسی کند و در صورت برقرار بودن ارتباط سطر به سطر شروع به فرستادن اطلاعات و حذف آنها از پایگاه داده بپردازد.

برای این کار یک تابع برای اتصال به پایگاه داده، یک تابع برای خواندن یک سطر از آن و یک تابع برای حذف سطر با توجه به primary key نوشته شده است. دقت کنید که primary key در جدول unsent_data ترکیب data_key و date است. (صرف نظر از مقدار خوانده شده، پارامتر خوانده شده در یک زمان مشخص یکتاست).

```
import requests
import psycopg2
import time

def connect_to_database():
    connection = None
    try:
        connection = psycopg2.connect(user = 'pi',
                                       password = '1234',
                                       host = 'localhost',
                                       port = '5432',
                                       database = 'data')

        cur = connection.cursor()

    except (Exception, psycopg2.DatabaseError) as error:
        print(error)

    return connection, cur

def select_from_database():
    sql = """SELECT * FROM unsent_data
            LIMIT 1"""
    connection, cur = connect_to_database()

    if connection is not None:
        cur.execute(sql)
        row = cur.fetchall()

        connection.commit()
        cur.close()
        connection.close()

    return row

def delete_from_database(data_key, date):
    sql = """ DELETE FROM unsent_data
            WHERE data_key = (%s) and date = (%s)"""
    connection, cur = connect_to_database()

    if connection is not None:
        cur.execute(sql, (data_key, date))

        connection.commit()
        cur.close()
        connection.close()
```


در ادامه تابع ckeck_server با فرستادن درخواست به 94.101.178.12/api/ping ارتباط با سرور را بررسی می‌کند.

```
def check_status_code(status_code):  
    if status_code == 200:  
        return True  
    else:  
        return False  
  
def check_server():  
    url = 'http://94.101.178.12/api/ping'  
    response = requests.get(url)  
    return check_status_code(response.status_code)  
  
f1 = open('./info/device_serial', 'r')  
device_serial = f1.read()  
f1.close()  
  
f2 = open('./info/private_key', 'r')  
private_key = f2.read()  
f2.close()
```

حالا کافیست هر یک ثانیه یک بار وضعیت اتصال به سرور بررسی شود و در صورت برقرار بودن ارتباط، سطر به سطر از پایگاه داده بخوانیم و در صورت ارسال موفقیت‌آمیز آنها را از پایگاه داده حذف کنیم.

```
def send_data(row):  
    url = 'http://94.101.178.12/api/post'  
  
    sensor_data = {  
        'deviceSerial': int(device_serial),  
        'privateKey' : private_key,  
        'timeInstant' : row[2],  
        'dataKey': row[0],  
        'dataValue' : row[1]}  
  
    result = requests.post(url, data = sensor_data)  
    if check_status_code(result.status_code):  
        delete_from_database(row[0], row[2])  
  
try:  
    while True:  
        while check_server():  
            row = select_from_database()  
            if len(row) != 0:  
                send_data(row[0])  
            else:  
                break  
  
        time.sleep(1)  
  
except KeyboardInterrupt:  
    pass
```

|

public_key_monitor.py

public_key کلیدی است که با در اختیار داشتن آن می‌توان از طریق اپ به سرور وصل شد و اطلاعات مربوط به آن را مشاهده کرد. از آن جا که public_key بر خلاف private_key ثابت نیست و ممکن است توسط کاربران تغییر داده شود، لازم است تا دائما محتویات فایل public_key خوانده شود و در صورتی که تغییر کرده بود به 94.101.178.12 اطلاعات جدید را بفرستد تا سرور نیز از این تغییر مطلع شود. دقت کنید که تابع peek_line طوری محتویات را می‌خواند که cursor جلو نرفته باشد و وقتی مجدد همان فایل را می‌خوانیم، باز هم public_key خوانده شود.

```
import time
import requests

def peek_line(f):
    position = f.tell()
    line = f.readline()
    f.seek(position)
    return line

f = open('./info/public_key', 'r')

f1 = open('./info/device_serial', 'r')
device_serial = f1.read()
f1.close()

f2 = open('./info/private_key', 'r')
private_key = f2.read()
f2.close()

prev_public_key = None

url = 'http://94.101.178.12/api/public_key_update'

try:
    while True:
        new_public_key = peek_line(f)
        if prev_public_key != new_public_key:
            update_data = {
                'deviceSerial': int(device_serial),
                'privateKey' : private_key,
                'publicKey' : new_public_key
            }
            print(update_data)
            update_result = requests.patch(url, data = update_data)
            print(update_result)
            print(update_result.text)

            prev_public_key = new_public_key
            time.sleep(1)
except KeyboardInterrupt:
    pass
```

- [1] <https://www.instructables.com/Raspberry-Pi-Tutorial-How-to-Use-the-DHT-22/>
- [2] <https://yantraas.com/send-sensor-data-from-arduino-to-raspberry-pi/>
- [3] <https://www.industrialshields.com/blog/raspberry-pi-for-industry-26/post/how-to-install-postgresql-in-raspberry-pi-plc-395>
- [4] <https://stackoverflow.com/questions/28253681/you-need-to-install-postgresql-server-dev-x-y-for-building-a-server-side-extensi>