**RPI RFID – IOTA**

**Appendix**

**RFID PIN DESCRIPTION:**

**VCC: supplies power for the module. This can be anywhere from 2.5 to 3.3 volts. You can connect it to 3.3V output from your Arduino. Remember connecting it to 5V pin will likely destroy your module!**

**GND: is the Ground Pin and needs to be connected to GND pin on the Arduino.**

**RST: is an input for Reset and power-down. When this pin goes low, hard power-down is enabled. This turns off all internal current sinks including the oscillator and the input pins are disconnected from the outside world. On the rising edge, the module is reset**.

**IRQ: is an interrupt pin that can alert the microcontroller when RFID tag comes into its vicinity.**

**MISO: pin acts as (Master-In-Slave-Out) when SPI interface is enabled, acts as serial clock when I2C interface is enabled and acts as serial data output when UART interface is enabled**.

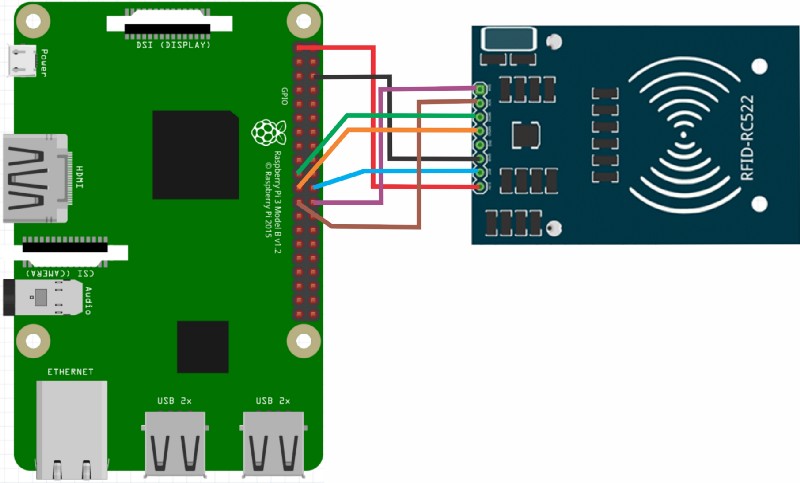
**MOSI:** **is SPI input to the RC522 module.**

**SCK: (Serial clock) accepts clock pulses provided by the SPI bus Master i.e. Arduino.**

**SDA: pin acts as Signal input when SPI interface is enabled, acts as serial data when I2C interface is enabled and acts as serial data input when UART interface is enabled. This pin is usually marked by encasing the pin in a square so it can be used as a reference for identifying the other pins.**

**Setting up the Raspberry pi configuration**

Firstly, connect the circuit as shown in below diagram



* **SDA** connects to **Pin 24**.
* **SCK** connects to **Pin 23**.
* **MOSI** connects to **Pin 19**.
* **MISO** connects to **Pin 21**.
* **GND** connects to **Pin 6**.
* **RST** connects to **Pin 22**.
* **3.3v** connects to **Pin 1**

After connections are made, open the raspberry pi terminal and run the following commands in the terminal.

* **sudo raspi-config**

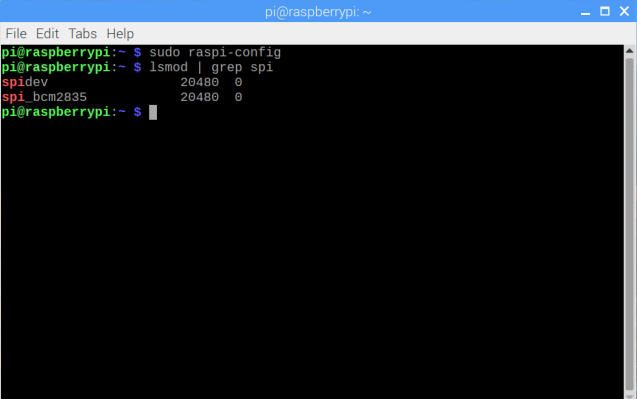
Type the above command in Raspberry pi terminal and go to “**interfacing options”** and navigate through **“p4 spi”** and enable the SPI Interface. After enable the SPI Interface then click on finish.

Before the SPI Interface is fully enabled we will first have to restart the Raspberry Pi. To Reboot the Raspberry pi type the following command in the terminal.

* **sudo reboot**

Once your Raspberry Pi has finished rebooting, we can now check to make sure that it has in fact been enabled. The easiest way to do this is to run the following command to see if **spi\_bcm2835** is listed.

* **lsmod | grep spi**



If you see **spi\_bcm2835**, then you can proceed on with this tutorial and skip on to the next section. If for some reason it had not appeared when you entered the previous command, try following the next three steps.

1. If for some reason the SPI module has not activated, we can edit the boot configuration file manually by running the following command on our Raspberry Pi.

* **sudo nano /boot/config.txt**

1. Within the configuration file, use **Ctrl + W** to find “**dtparam=spi=on**“.

If you have found it, check to see if there is a **#** in front of it. If there is, remove it as this is

commenting out the activation line. If you can’t find the line at all, add “**dtparam=spi=on**”

to the bottom of the file.

Once you have made the changes, you can press **Ctrl + X** then pressing **Y** and then **Enter** to

save the changes.

1. You can now proceed from **sudo reboot** again, rebooting your Raspberry Pi then checking to see if the module has been enabled.

**Getting Python Ready for the RFID RC522**

1. Before we start programming, we first need to update our Raspberry Pi to ensure it’s running the latest version of all the software. Run the following two commands on your Raspberry Pi to update it.

* **sudo apt-get update**
* **sudo apt-get upgrade**

1. Now the final thing we need before we can proceed is to install **python3-dev**, **python-pip** and git packages. Simply run the following command on your Raspberry Pi to install all of the required packages for this guide on setting up your RFID reader.

* **sudo apt-get install python3-dev python3-pip**

**Clone the Python Library SPI Py and install it to your Raspberry Pi to interact with the RFID RC522**.

**cd ~**

**git clone https://github.com/lthiery/SPI-Py.git**

**cd ~/SPI-Py**

**sudo python setup.py install**

**cd ~**

**git clone https://github.com/pimylifeup/MFRC522-python.git**

1. To begin, we must first install the Python Library spidev to our Raspberry Pi using the python “**pip**” tool that we downloaded in the previous step.

Run the following command on your Raspberry Pi to install **spidev** to your

Raspberry Pi through pip.

Please note that we use sudo here to ensure that the package is installed so that all

users can utilize it and not just the current user.

* **sudo pip3 install spidev**

1. Now that we have installed the **spidev** library to our Raspberry Pi we can now proceed to installing the MFRC522 library using pip as well.

* **sudo pip3 install mfrc522**

1. With the library now saved to our Raspberry Pi, we can begin programming for our RFID RC522.

**Here we want to install two more libraries i.e.… PyOTA library & pretty table**

**PyOTA Installation:**

To install the latest version run the following command in RPI terminal

* **pip install pyota**

Optional C Extension

PyOTA has an optional C extension that improves the performance of its cryptography features significantly (speedups of **60x**are common!).

To install this extension, use the following command:

* **pip install pyota[ccurl]**

**pretty table installation:**

To install the pretty table library, run the following command in RPI terminal

* **pip install PrettyTable**

**Programming with RFID**

Here we are using two codes one is for Recording the data and another one is for retrieving the data.

**First code:**

The following Python code is the part where we capture the tagID from the RFID reader, together with the hotel name and room number, and sends it to the tangle.

Copy and paste the following code in MFRC522-python folder.

The path of the folder is /home/pi/MFRC522-python.

Here we are using python script so you should save the file as .py format.

Source code: Reset\_register.py

**# Import datetime library**

**from datetime import datetime**

**# Import GPIO library**

**import RPi.GPIO as GPIO**

**# Import simplified version of the MFRC522 library**

**import SimpleMFRC522**

**# Import the PyOTA library**

**import iota**

**# Import json**

**import json**

**# Define IOTA address where all transactions (cleaning records) are stored, replace with your own address.**

**# IOTA addresses can be created with the IOTA Wallet**

**CleaningLogAddr = b"paste your IOTA log address here"**

**# Create IOTA object, specify full node to be used when sending transactions.**

**# Notice that not all nodes in the field.deviota.com cluster has enabled attaching transactions to the tangle**

**# In this case you will get an error, you can try again later or change to a different full node.**

**api = iota.Iota("https://nodes.thetangle.org:443")**

**# Define static variable**

**hotel = "Hotel IOTA View"**

**# Create RFID reader object**

**reader = SimpleMFRC522.SimpleMFRC522()**

**# Main loop, executes when an RFID tag (ID card) is close to the reader**

**try:**

**while True:**

**# Show welcome message**

**print("\nWelcome to the Hotel IOTA cleaning log system")**

**print("Press Ctrl+C to exit the system")**

**# Get room number**

**room\_number = input("\nPlease type room number and press Enter: ")**

**print("\nThank you, now hold your ID card near the reader")**

**# Get card ID from the reader**

**id, text = reader.read()**

**# Create json data to be uploaded to the tangle**

**data = {'tagID': str(id), 'hotel': hotel, 'room\_number': room\_number}**

**# Define new IOTA transaction**

**pt = iota.ProposedTransaction(address = iota.Address(CleaningLogAddr),**

**message = iota.TryteString.from\_unicode(json.dumps(data)),**

**tag = iota.Tag(b'HOTELIOTA'),**

**value = 0)**

**# Print waiting message**

**print("\nID card detected...Sending transaction...Please wait...")**

**# Send transaction to the tangle**

**FinalBundle = api.send\_transfer(depth=3, transfers=[pt], min\_weight\_magnitude=14)['bundle']**

**# Print confirmation message**

**print("\nTransaction sucessfully completed, have a nice day")**

**# Clean up function when user press Ctrl+C (exit)**

**except KeyboardInterrupt:**

**print("cleaning up")**

**GPIO.cleanup()**

Save the code as Reset\_register.py in MFRC522-python folder.

Second code:

The following Python code is used by the receptionist when retrieving and displaying the Reset log.

**# Imports from the PyOTA library**

**from iota import Iota**

**from iota import Address**

**from iota import Transaction**

**from iota import TryteString**

**# Import json library**

**import json**

**# Import datetime libary**

**import datetime**

**# Import from PrettyTable**

**from prettytable import PrettyTable**

**# Define IOTA address where all transactions (cleaning records) are stored, replace with your own address.**

**address = [Address(b'paste your IOTA address here')]**

**# Define full node to be used when retrieving cleaning records**

**iotaNode = "https://nodes.thetangle.org:443"**

**# Create an IOTA object**

**api = Iota(iotaNode)**

**# Create PrettyTable object**

**x = PrettyTable()**

**# Specify column headers for the table**

**x.field\_names = ["tagID", "Hotel", "Room", "last\_cleaned"]**

**# Find all transacions for selected IOTA address**

**result = api.find\_transactions(addresses=address)**

**# Create a list of transaction hashes**

**myhashes = result['hashes']**

**# Print wait message**

**print("Please wait while retrieving cleaning records from the tangle...")**

**# Loop trough all transaction hashes**

**for txn\_hash in myhashes:**

**# Convert to bytes**

**txn\_hash\_as\_bytes = bytes(txn\_hash)**

**# Get the raw transaction data (trytes) of transaction**

**gt\_result = api.get\_trytes([txn\_hash\_as\_bytes])**

**# Convert to string**

**trytes = str(gt\_result['trytes'][0])**

**# Get transaction object**

**txn = Transaction.from\_tryte\_string(trytes)**

**# Get transaction timestamp**

**timestamp = txn.timestamp**

**# Convert timestamp to datetime**

**clean\_time = datetime.datetime.fromtimestamp(timestamp).strftime('%Y-%m-%d %H:%M:%S')**

**# Get transaction message as string**

**txn\_data = str(txn.signature\_message\_fragment.decode())**

**# Convert to json**

**json\_data = json.loads(txn\_data)**

**# Check if json data has the expected json tag's**

**if all(key in json.dumps(json\_data) for key in ["tagID","hotel","room\_number"]):**

**# Add table row with json values**

**x.add\_row([json\_data['tagID'], json\_data['hotel'], json\_data['room\_number'], clean\_time])**

**# Sort table by cleaned datetime**

**x.sortby = "last\_cleaned"**

**# Print table to terminal**

**print(x)**

save this file as Reset\_log.py

**Running the python files in RPI Terminal**

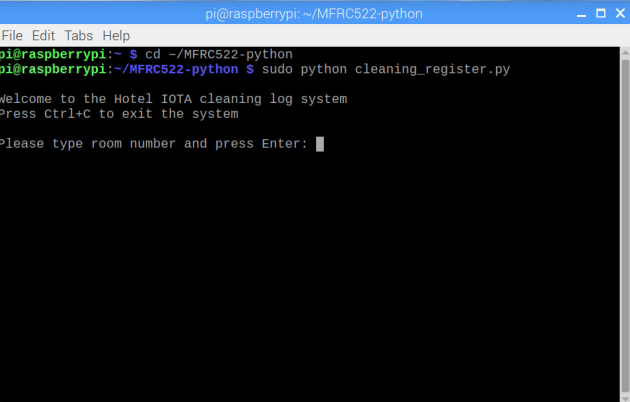
* To run the cleaning\_register.py follow the below steps

First go to directory where the file is saved.

**cd ~/MFRC522-python**

Then type **sudo python Reset\_registry.py** to run the program.

Once you run the python file the following tab appears



Enter the room number and press the Enter button

Then keep the tagID near the sensor to read the data.

The resultant sensor data was seen in Trinity wallet.

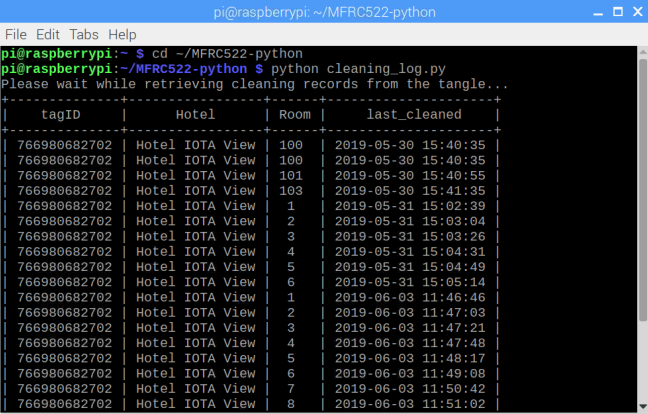
* To run the cleaning\_log.py follow the below steps

First go to directory where the file is saved.

**cd ~/MFRC522-python**

Then type **python Reset\_log.py** to run the program**.**

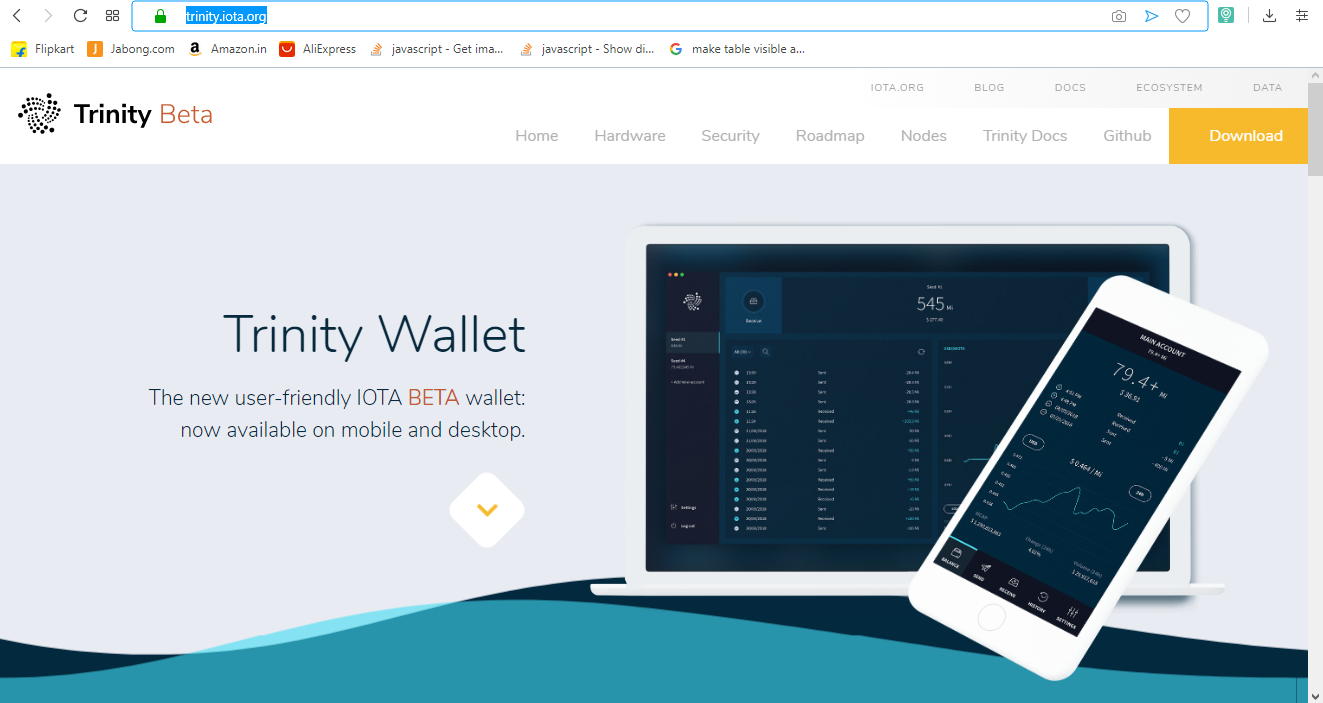
**Output for the Reset\_log.py is shown in below**

****

**You need to install RSON 0.9 using** pip install rson==0.9 to view the data in terminal.

**Setting Up the Trinity Application**

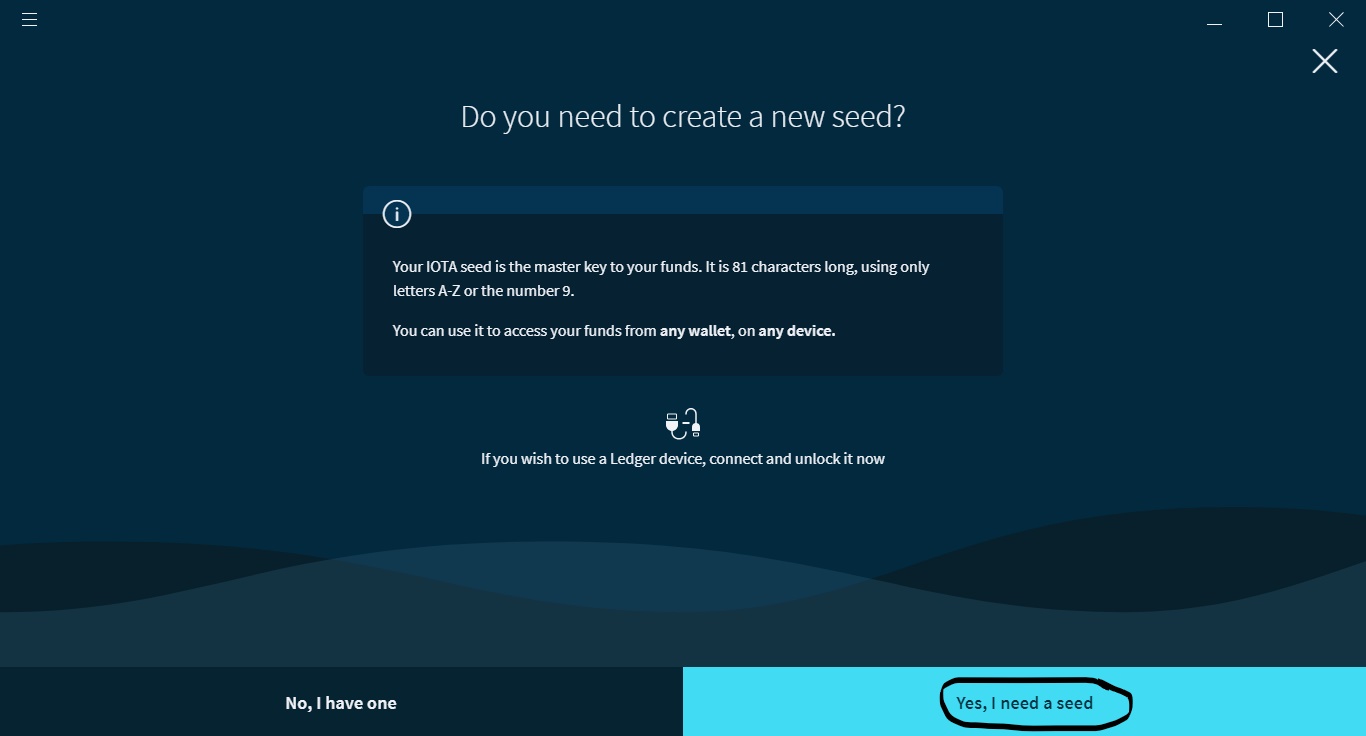
Go to <https://trinity.iota.org>



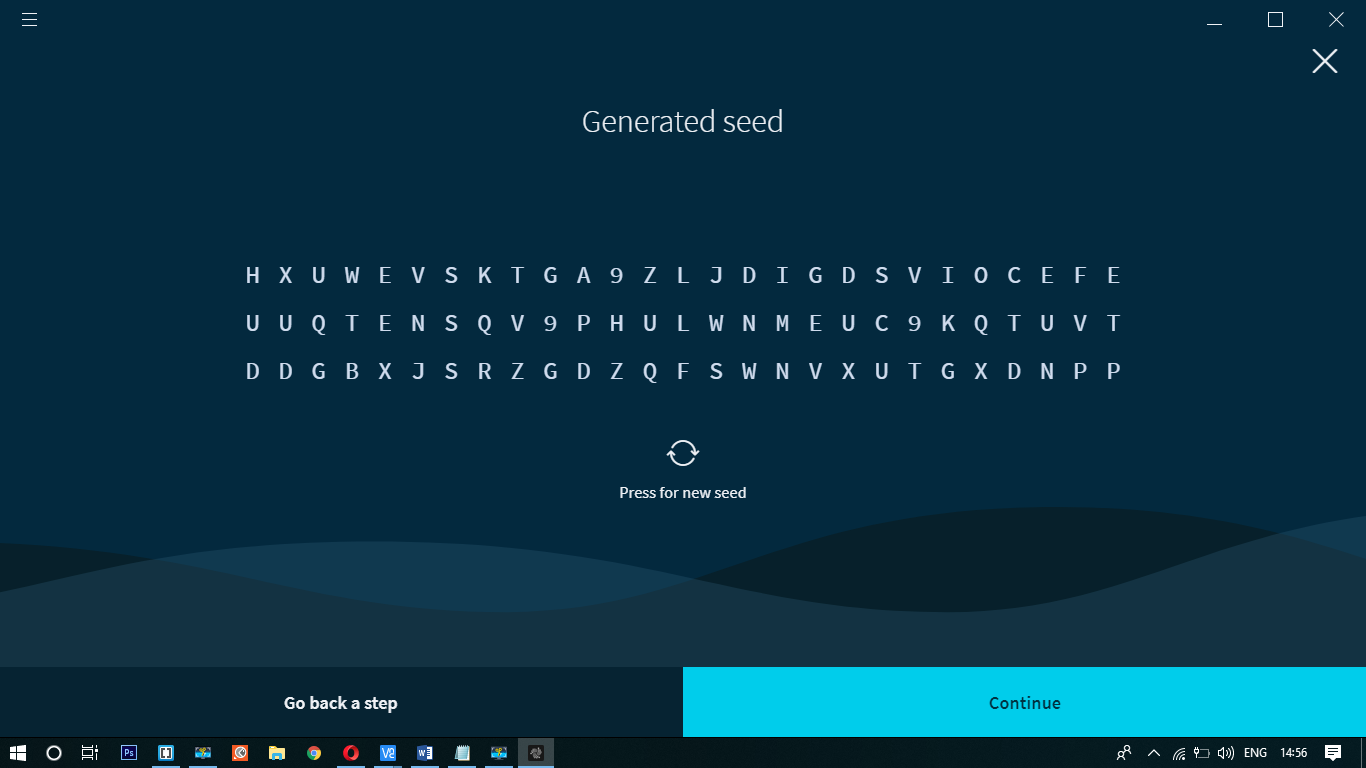
Go to Downloads and Download and install the trinity app.

After installation was done open the app and login in to the app.

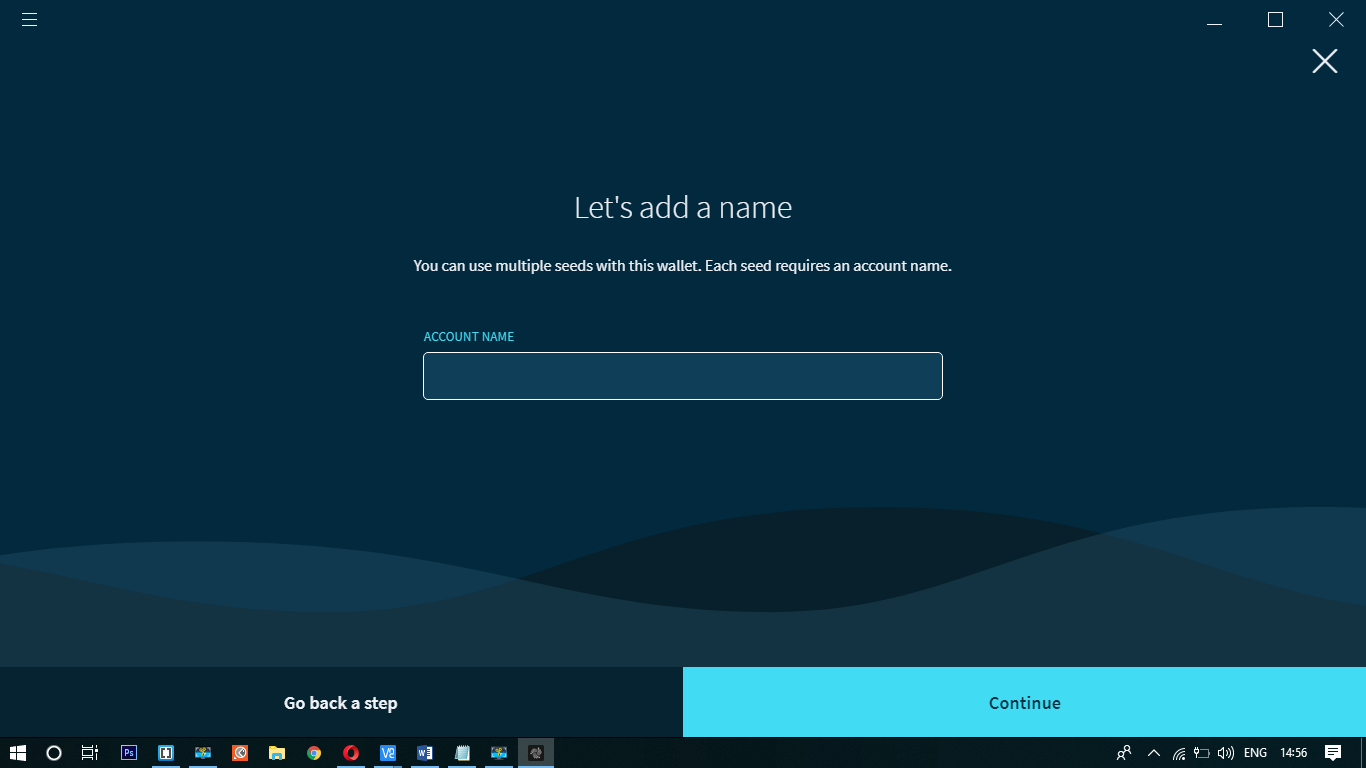
After login we have to create a “seed”



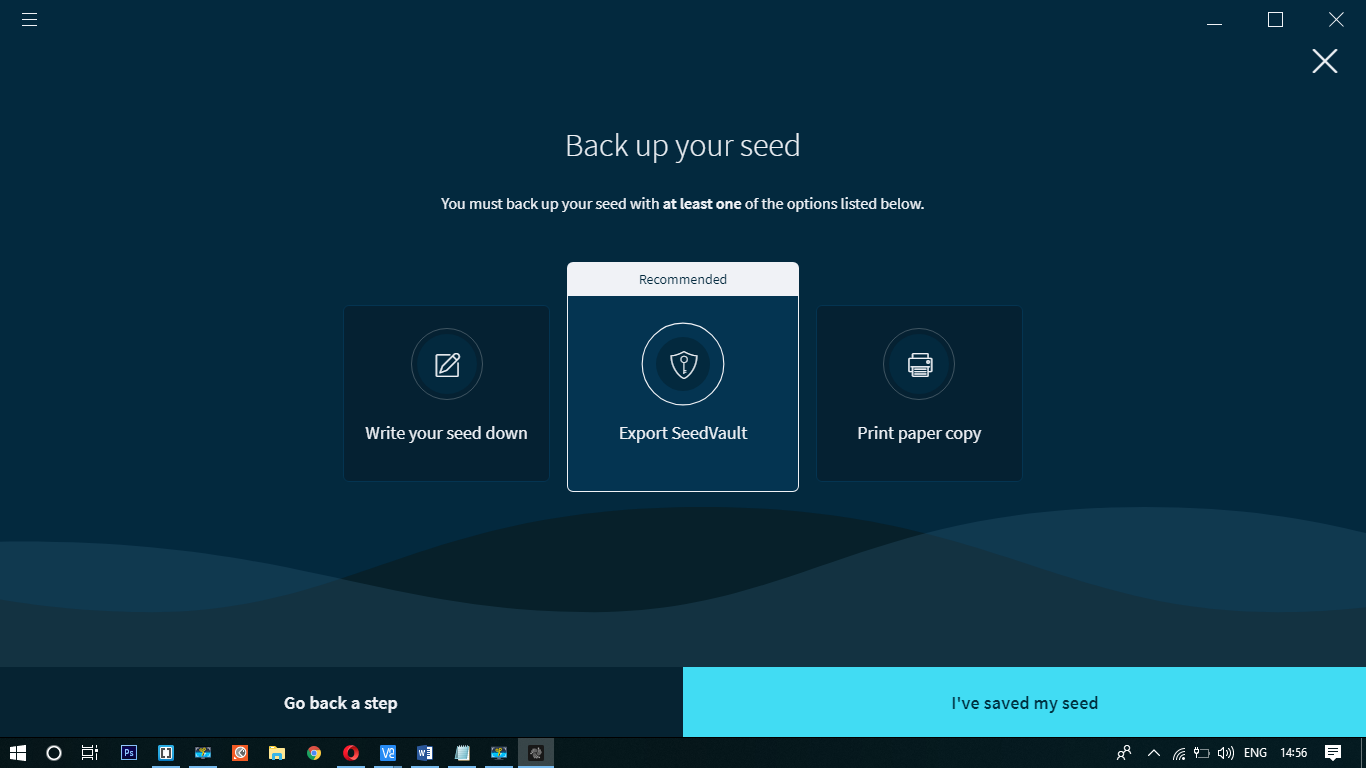
Click on Yes,I need a seed button as shown in above. Then the following tab appears. Here you want to select 10 letters and click on continue



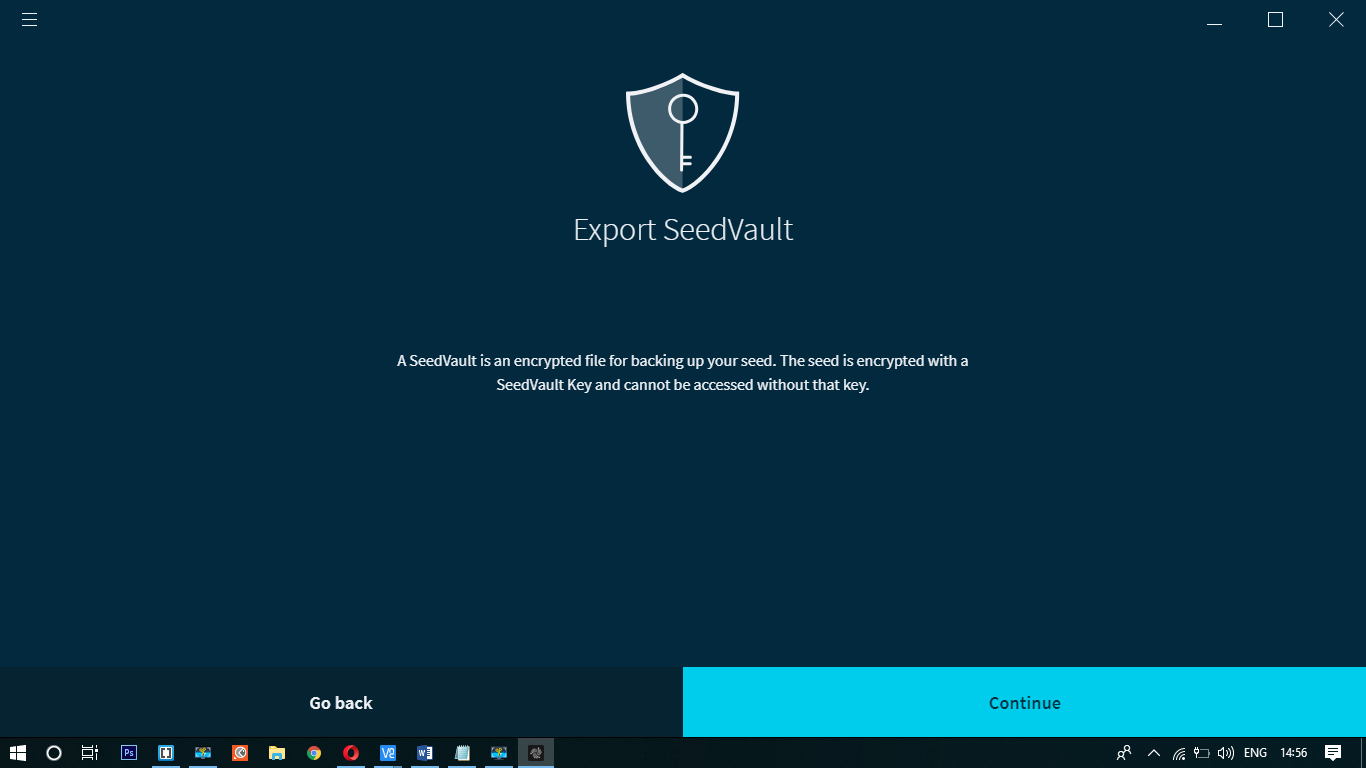
Then add your account name and click on continue



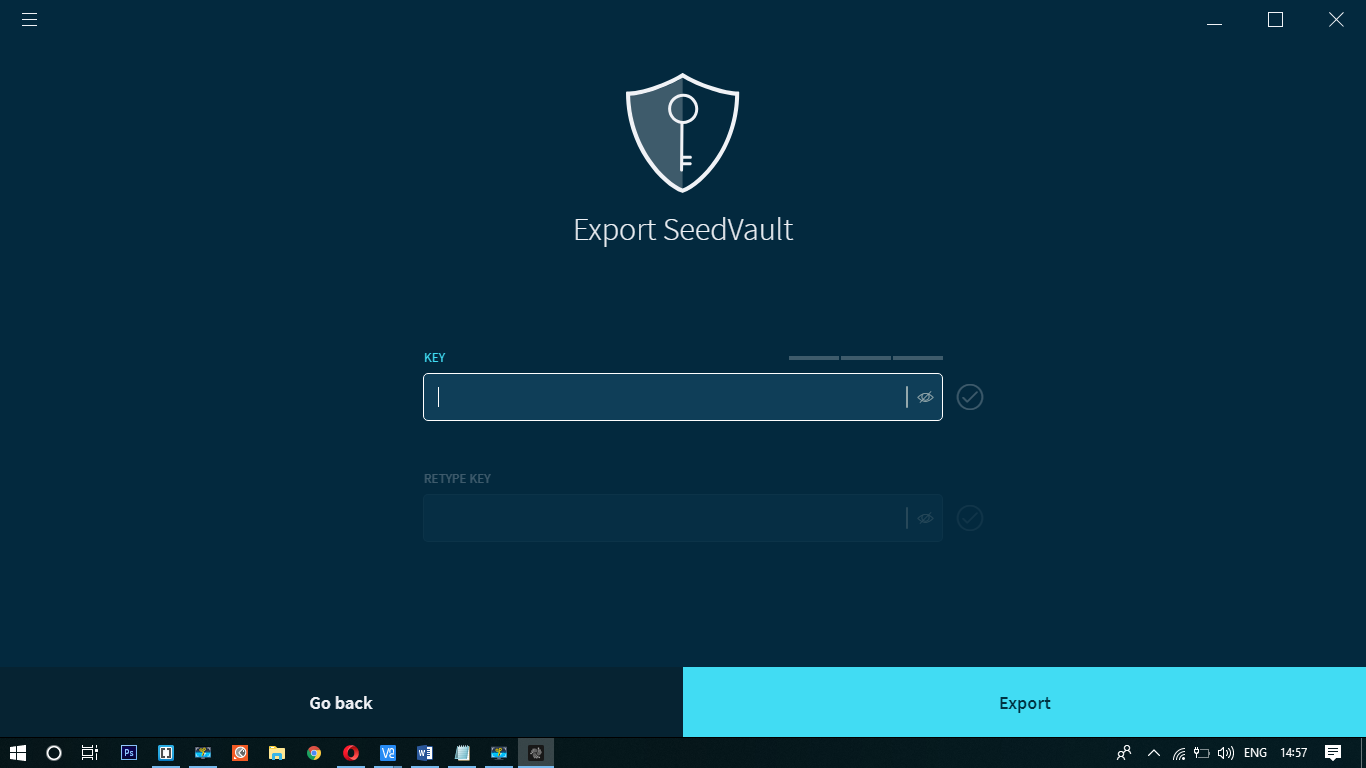
Here you have to backup you seed. Click on Export seed vault.

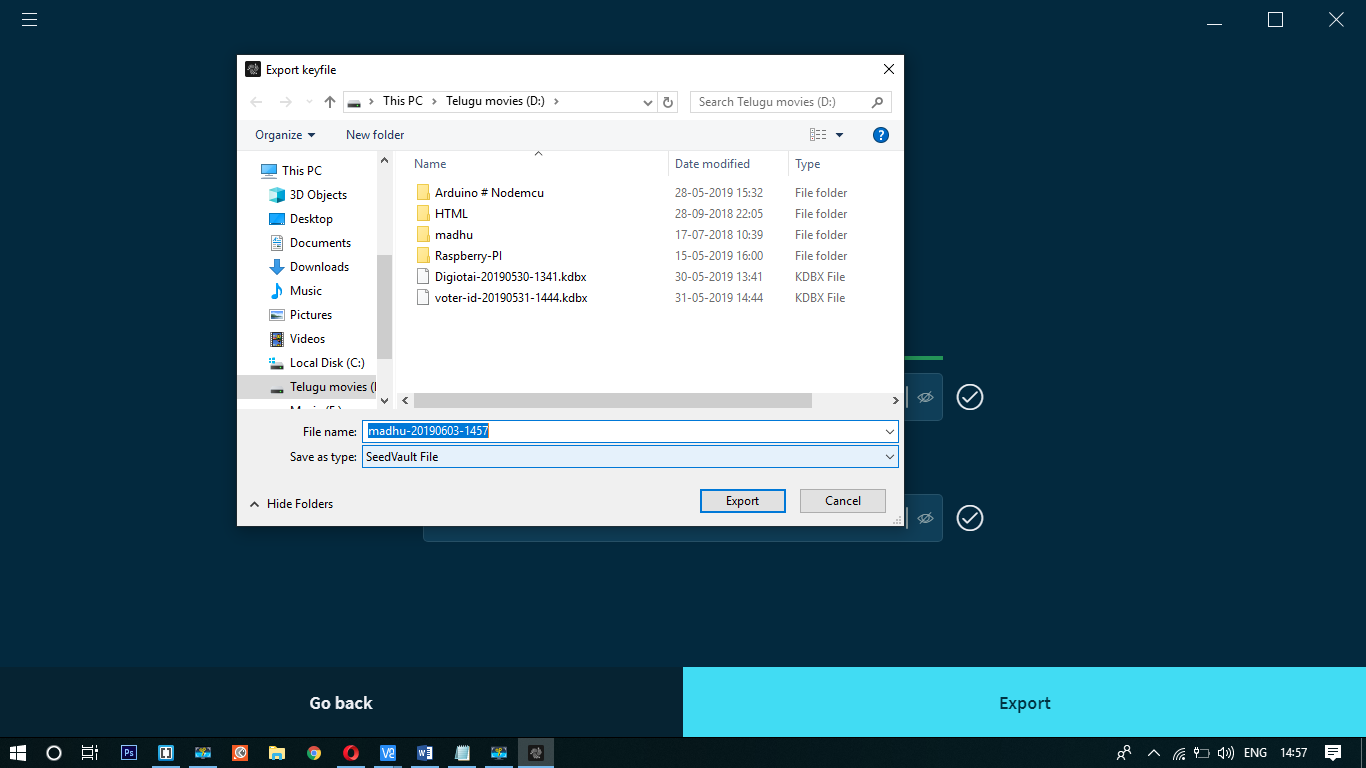


After click on Export seed vault the following tab appears. Then press click on continue for two more steps.



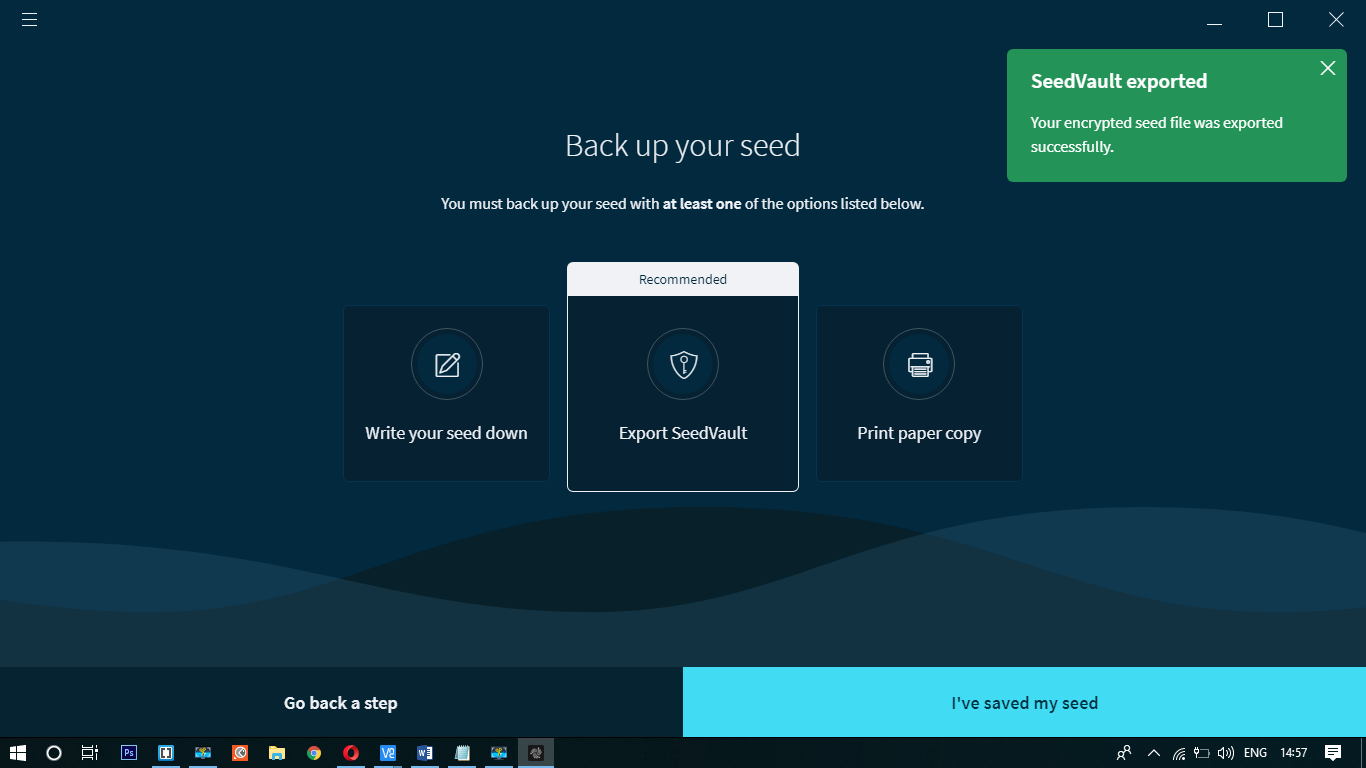
Then you have to create a key for your seed



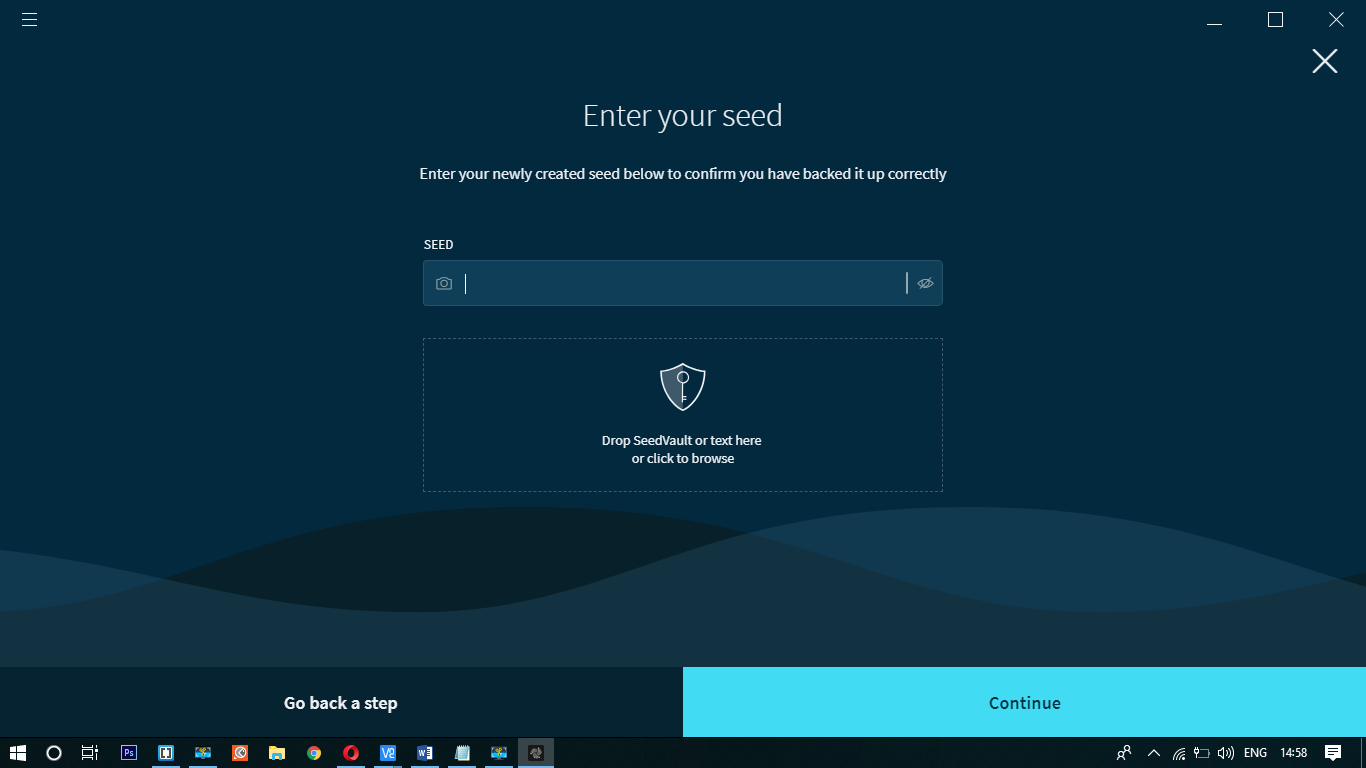
Enter the key and click on Export button. By clicking on export button, you will be asked to save a file. 

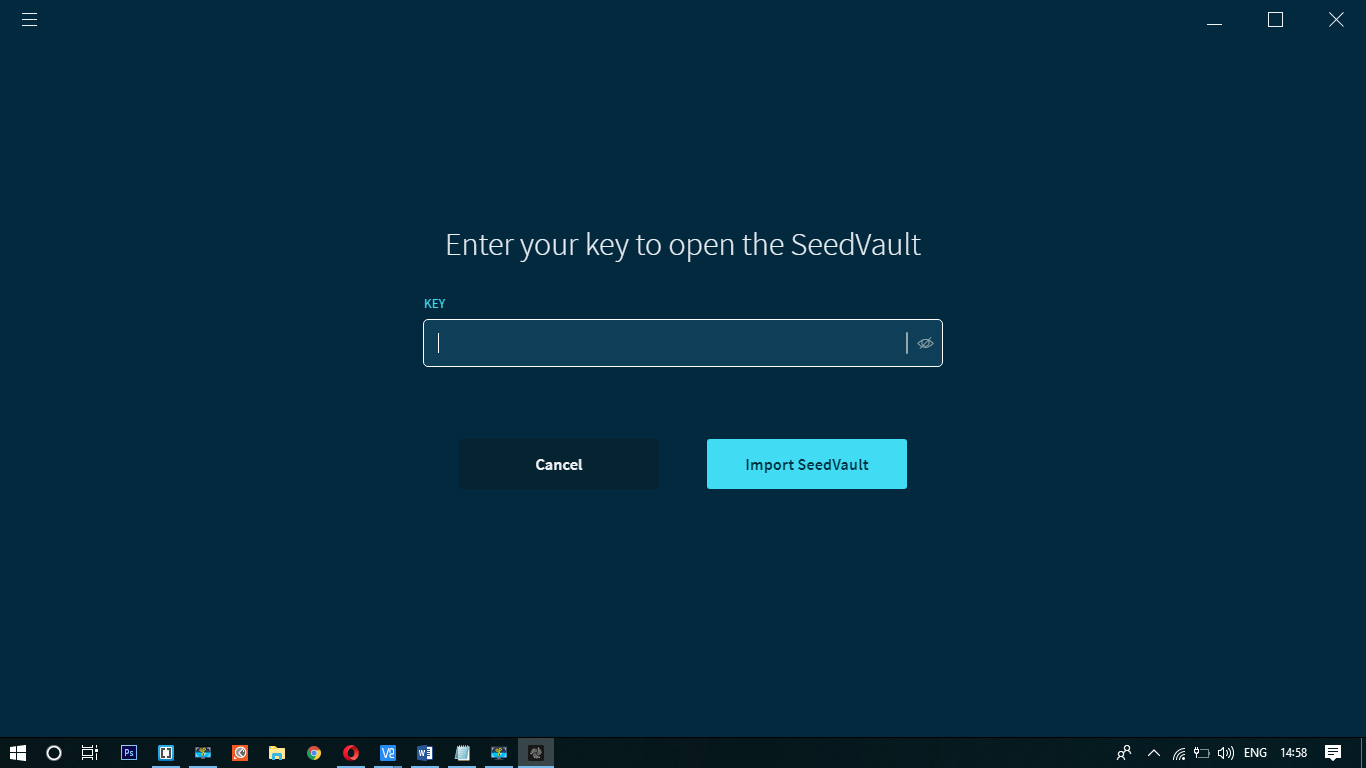
Save the file

Then click on I’have saved my seed button

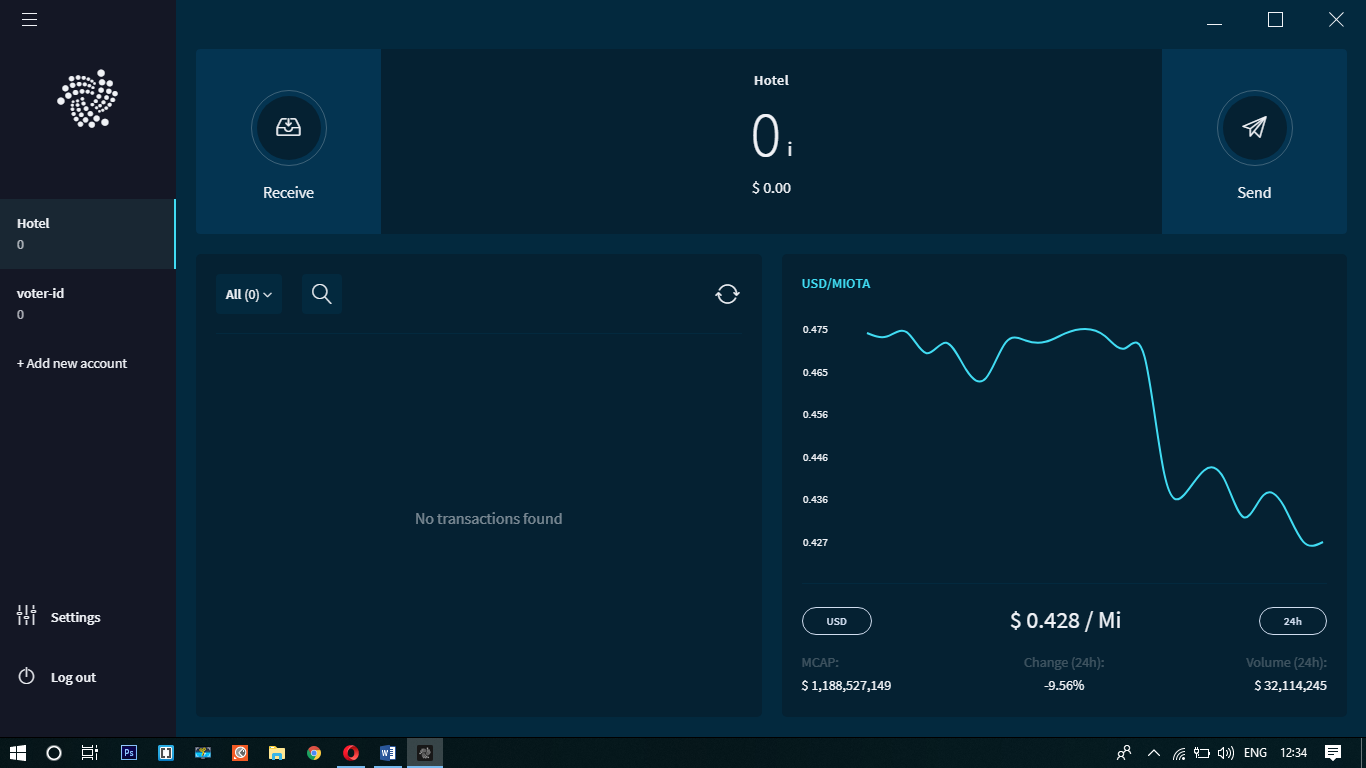


Here you have to enter or upload your seed which was saved before. Upload your seed and click on continue





Then enter your key and click on import seedvault. Here you will see a monitoring the output values as shown in below.



Note: you have to replace the IOTA log address in code. You can find your log address in your seed. Click on seed settings -> view addresses here you will find your address copy and paste in your code