University of Malta

Master of Science in Blockchain and Distributed Ledger Technologies

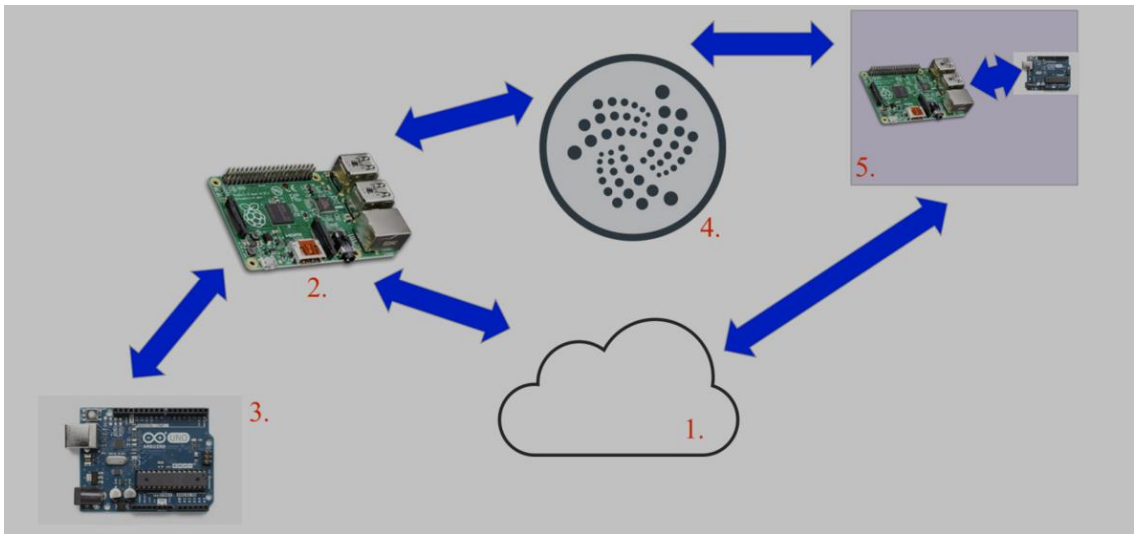
DLT5403: DLT and the Internet of Things

Assignment Part 2 - IOTA

Date Submitted: 15th September 2023
Group: Karsten Guenther
Lecturer: Prof. Joshua Ellul

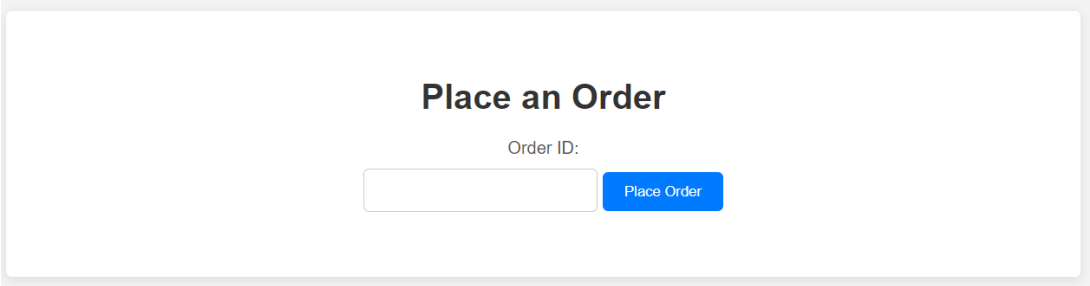
Architecture Overview:

Consisting of a Packager Bot, IOTA ledger, Lock Device, Marketplace Front-end, Warehouse Controller, and Home Controller, this system empowers users to order deliveries, tracks courier movements, and provides secure verification techniques for IoT devices and lightweight public-key protocols. Furthermore, the system can enhance real-time private authentication and strives for increases in energy efficiency, sustainability, user satisfaction, and environmental responsibility.
The description of the components of the system is given below:



Market Place Frontend:

A basic frontend was implemented to initiate a delivery. The idea is that if a user wants an order delivered when he/she is not at home. They can simply pass a valid order ID to this front end and initiate the delivery.



Warehouse Controller:

The warehouse controller is responsible for handling orders from the marketplace front end and assigning them to the available packaging bot. Then after packaging is done the controller sends information to the IOTA private key and sends this information to the driver and user.

The warehouse controller is written in Python language and it uses the following Python modules:

- Iota_sdk for communicating the IOTA.
- Flask to build a web server.
- Requests to call the APIs.
- PySerial module for communicating with the Packaging bot.

_client_address.py | iota_create_client_account.py | iota_client_address_balance.py | send_iota_client_to_distributor

arduino_serial_interface.py > ...

```python
import serial

ser = serial.Serial('COM5', 9600)

while True:
    if(ser.in_waiting > 0):
        while ser.in_waiting > 0:
            print((ser.read().decode()), end="")
        print()
```

IOTA_PROJECT

- __pycache__
- client-walletdb
- distributor-walletdb
- arduino_serial_interface.py
- client.stronghold
- config.json
- distributor.stronghold
- iota_client_address_balance.py
- iota_create_client_account.py
- iota_create_client_address.py
- iota_create_client_mnemonic.py
- iota_create_dist_account.py
- iota_create_dist_address.py
- iota_create_dist_mnemonic.py
- iota_dist_address_balance.py
- rw_json_file.py
- send_iota_client_to_distributor.py
- test.py

send_iota_client_to_distributor.py > ...

```python
from iota_sdk import Client, MnemonicSecretManager, AddressAndAmount,
import rw_json_file

data = rw_json_file.read("config.json")

node_url = data['NODE_URL']

# Create a Client instance
client = Client(nodes=[node_url])

secret_manager = MnemonicSecretManager(
    data['CL_MNEMONIC'])

address_and_amount = AddressAndAmount(
    1000000,
    data["DISTRIBUTOR_ADDRESS"],
)

# Create and post a block with a transaction
block = client.build_and_post_block(secret_manager, output=address_and
print(f'Block sent: {data["EXPLORER_URL"]}/block/{block[0]}')
```

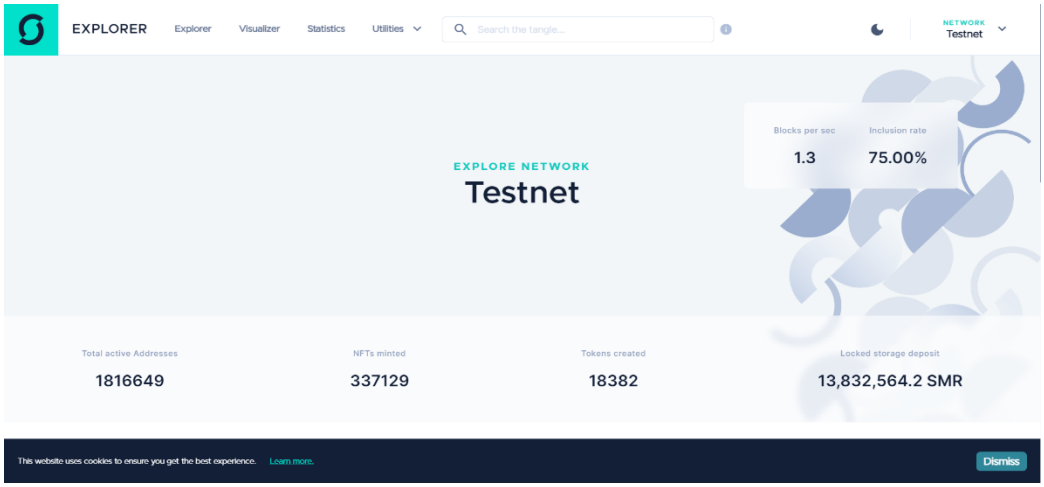
```python
from iota_sdk import Wallet
import json
import rw_json_file

data = rw_json_file.read("config.json")
wallet = Wallet(data['CL_WALLET_DB_PATH'])

account = wallet.get_account('ClientAccount')

# Sync account with the node
_balance = account.sync()

# Just calculate the balance with the known state
balance = account.get_balance()
print(f'Balance {json.dumps(balance.as_dict(), indent=4)}')
```

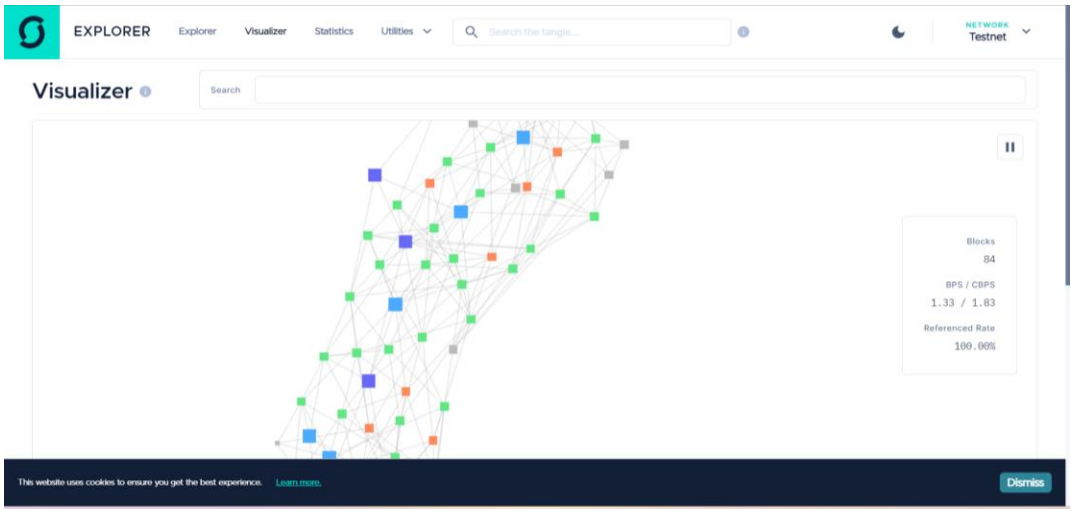
Packaging Bot:

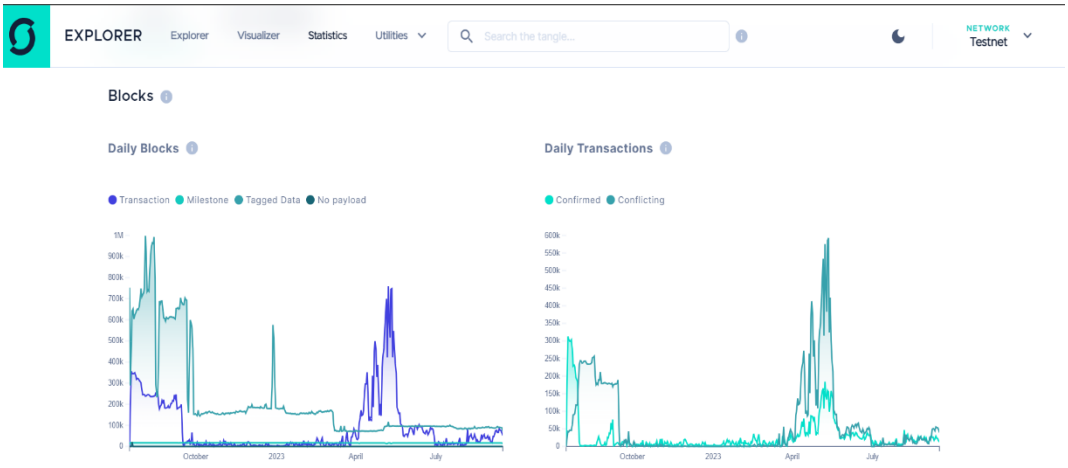
The packaging bot is implemented using Arduino UNO. These bots simply connect to the server using a USB port. The server detects if there are some bots connected to it. Then it assigns an order to them then in response the bot gives a package ID.

IOTA:

Details of transactions are meticulously logged onto a secure, decentralized ledger. Using the warehouse's IOTA private key, the Order Numbers, Package IDs, and digitally signed information are stored by the Warehouse Controller.



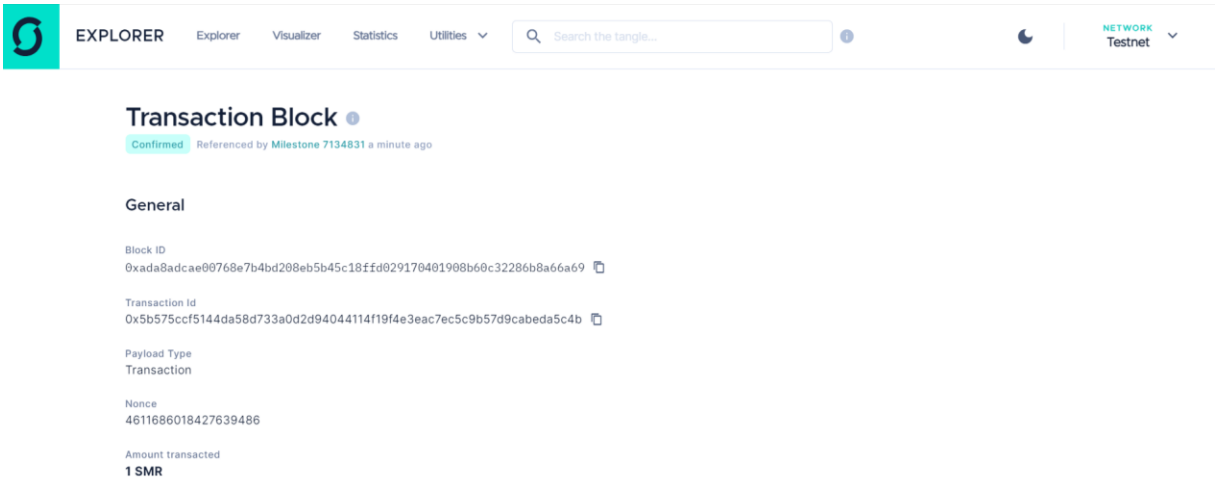




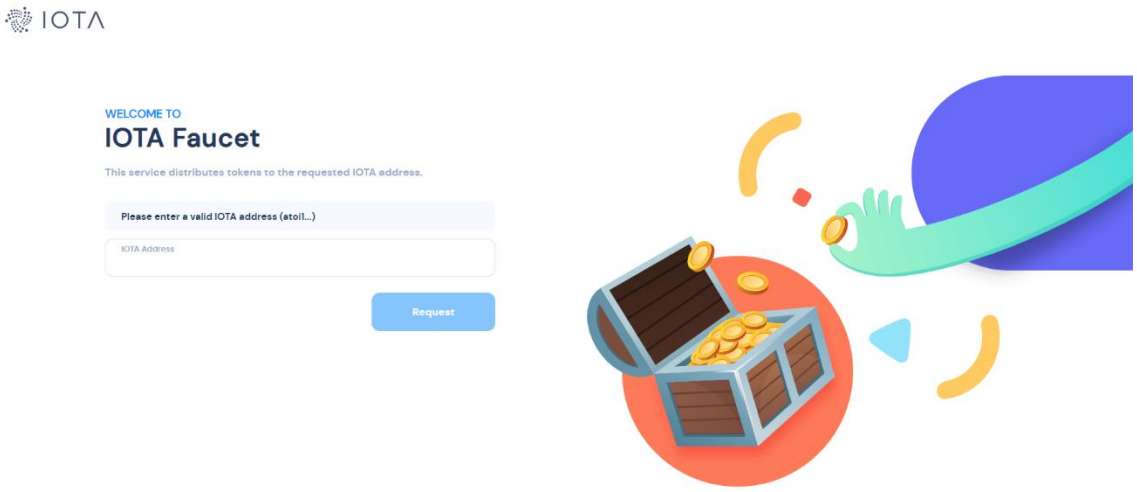
Home Controller:

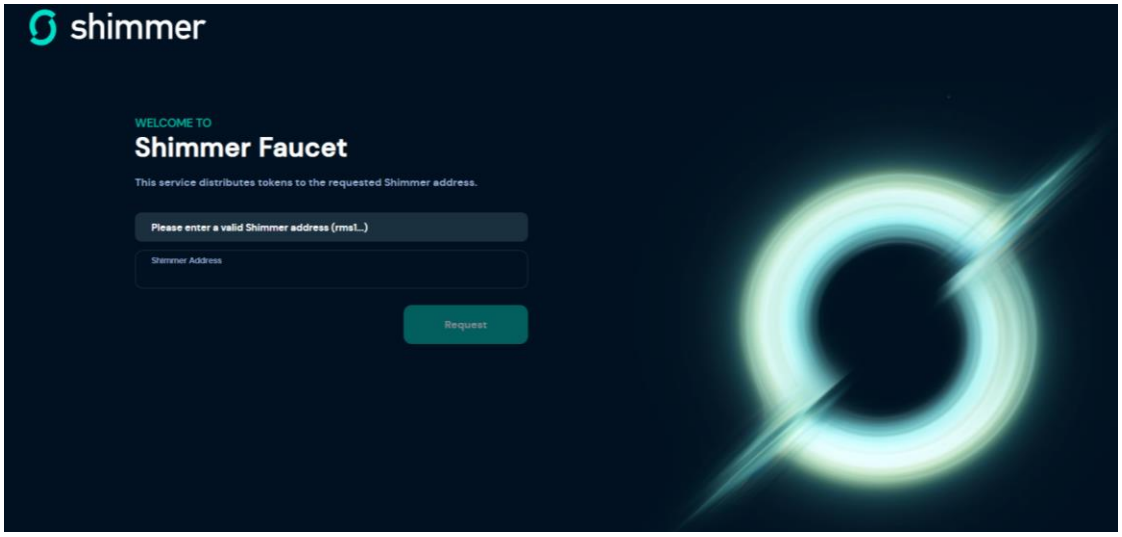
It is responsible for communicating with the distributor server, driver, and the knock lock. The distributor sends no. of knocks to the home controller and the driver the driver knocks the knock lock for the same number and the home controller verifies it. After the verification is done the payment is made from the client's wallet to the distributor's IOTA wallet.

The home controller is written in Python and it uses the pyserial, Iota_sdk, flask, and requests module. The below snippet shows a transaction of 1 IOTA from the client’s wallet to distributor wallet on the IOTA explorer.



The IOTA provides test tokens for testing purposes we don't need to buy any Tokens for testing. We can get tokens from the IOTA faucet.



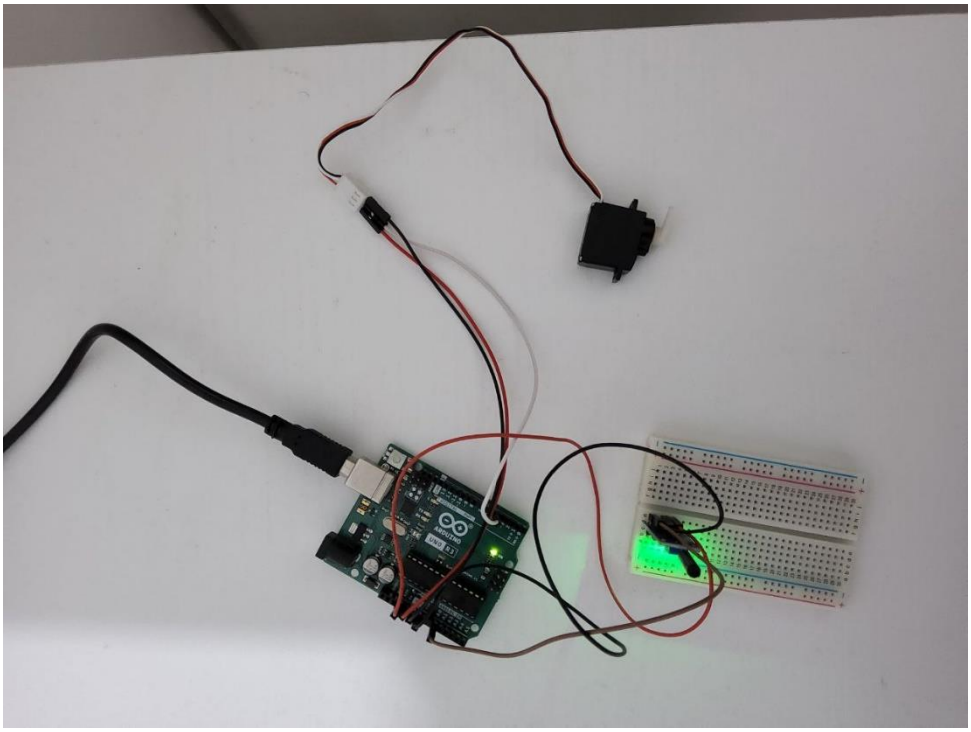


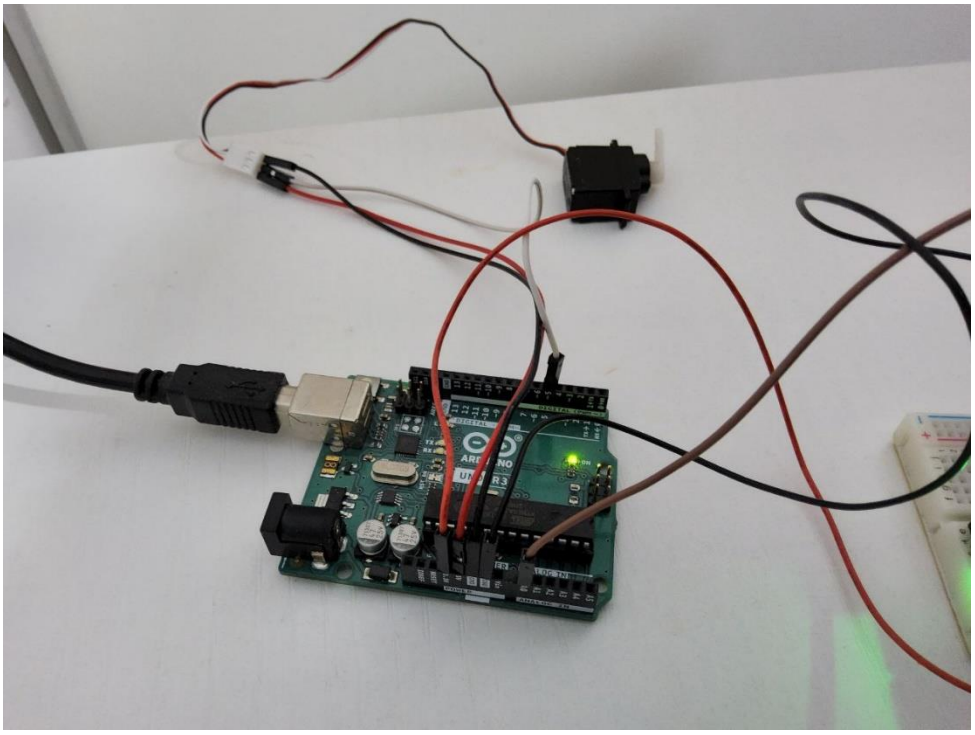
Lock Device:

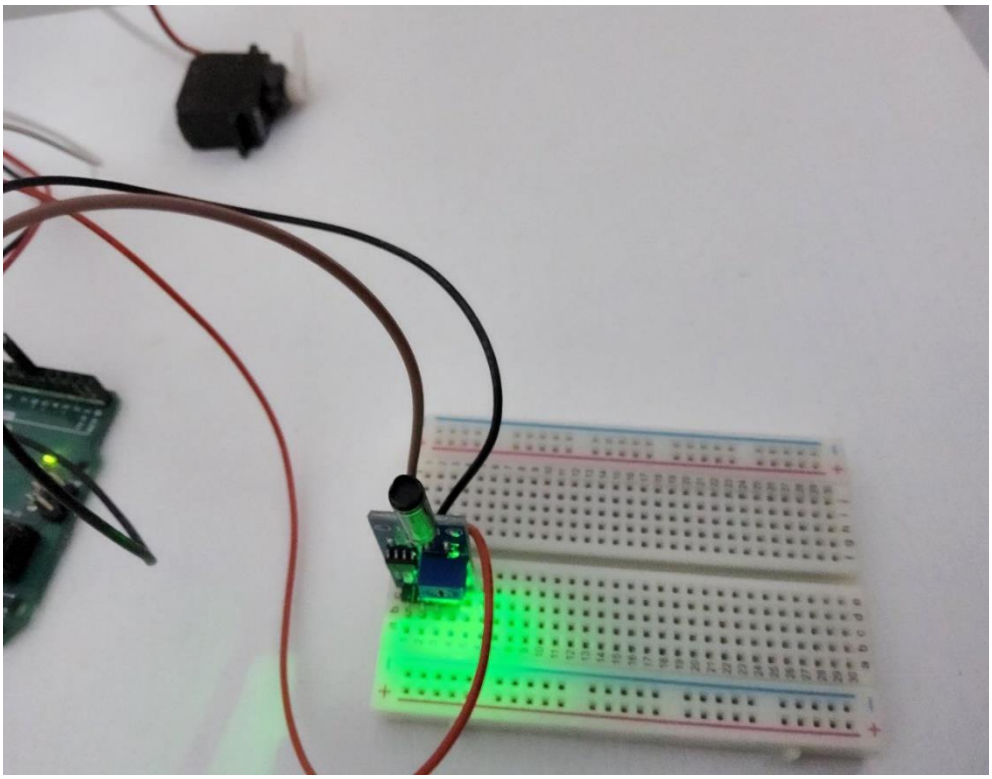
The lock device is an Arduino-based device that consists of an Arduino UNO and a vibration sensor. It detects no. of knocks by the driver and sends it to the connected home controller system through a USB port then the home controller compares it with knocks sent from the distributor server.

```
#include <Servo.h>

Servo myservo;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  myservo.attach(5);
  myservo.write(0);
}

uint8_t knockCount = 0;
unsigned long previousKnock = 0;
void loop() {
  int PiezoValue = analogRead( A0 );
  //Serial.println(PiezoValue);
 //two kknocks
  if (PiezoValue < 250){
    Serial.println("Knock");
    unsigned long currentKnock = millis();
    if(currentKnock - previousKnock > 5000){
      knockCount = 0;
    }
    knockCount++;
    if(knockCount >= 3){
      knockCount = 0;
      Serial.println("3 Knocks");
      myservo.write(90);
      delay(5000);
```

```
Done compiling.
Sketch uses 3250 bytes (10%) of program storage space. Maximum is 32256 bytes.
Global variables use 248 bytes (12%) of dynamic memory, leaving 1800 bytes for local variables. Maximum is 2048 bytes.
```







Checklist:

1. Initiate delivery from frontend [frontend done]
2. Communicate with packaging bot (arduino) [pending]
3. Send package details to client and driver [pending]
4. Driver knocks arduino [knock lock done]
5. After verifying knock send IOTA token from client account to distributor [done]

Improvements to The System:

- Use wireless Protocols like WiFi or BLE for communication with the packaging bot.
- The front end should be improved by automating fetching the order details from the server. So, the user does not need to fill in the order ID.
- The driver device and the main controller should use some kind of sensor to communicate with each other rather than driver knocking the lock device.