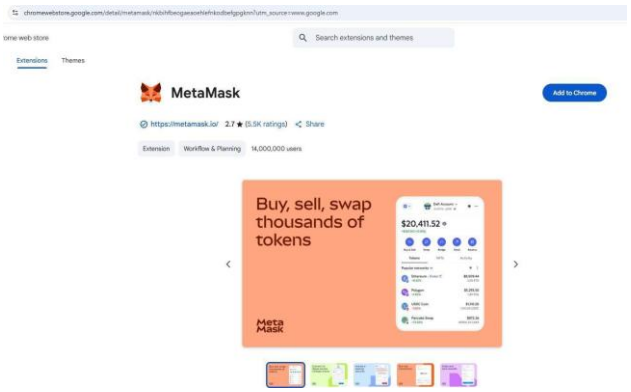
# BAB IV
# IMPLEMENTASI DAN PEMBAHASAN

## 4.1. Implementasi dan Uji Coba Sistem

Bagian ini menjelaskan proses implementasi sistem distribusi *Airdrop* menggunakan *Smart contract* berbasis jaringan *Monad Testnet*, yang disusun berdasarkan rancangan pada Bab III. Implementasi melibatkan beberapa tahapan utama: konfigurasi dompet digital (*wallet*), penulisan dan kompilasi *Smart contract*, serta pengujian distribusi *token* ke banyak alamat secara otomatis.

### 4.1.1 Konfigurasi *Wallet* dan Jaringan

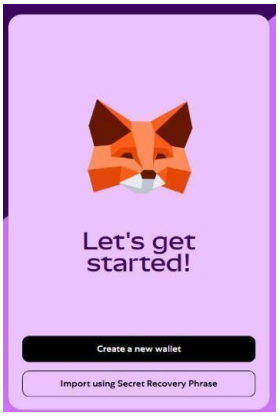
1. Langkah pertama adalah menginstal ekstensi *MetaMask* di browser,




**Gambar 4.1 Menginstal ekstensi *MetaMask***

Gambar ini menunjukkan langkah awal instalasi ekstensi *MetaMask* di peramban (*browser*). Proses ini penting karena *MetaMask* digunakan sebagai *wallet* untuk menghubungkan akun pengguna dengan jaringan *Monad Testnet*.

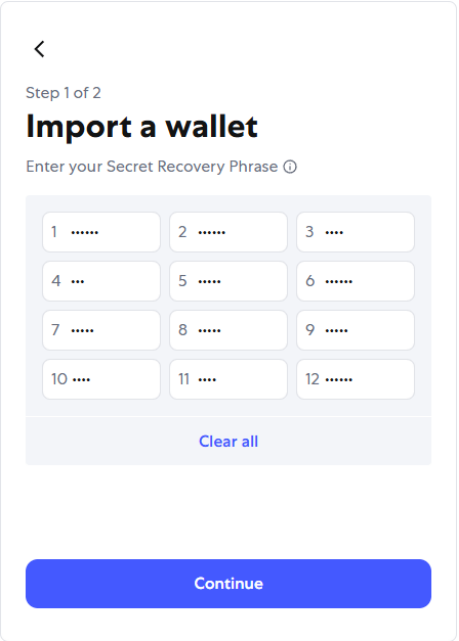
2. kemudian membuat atau mengimpor *wallet* yang akan digunakan.




**Gambar 4.2 Membuat atau mengimpor *wallet***

Gambar ini menampilkan antarmuka *MetaMask* pada langkah pembuatan *wallet.* Pengguna diberikan pilihan untuk membuat *wallet* baru dengan kata sandi atau mengimpor *wallet* yang sudah ada menggunakan frasa pemulihan (*Secret Recovery Phrase).* Langkah ini memungkinkan pengguna untuk memulihkan *wallet* lama atau menyiapkan *wallet* baru yang selanjutnya akan digunakan untuk terhubung ke jaringan *Monad Testnet* dalam proses distribusi *Airdrop.*

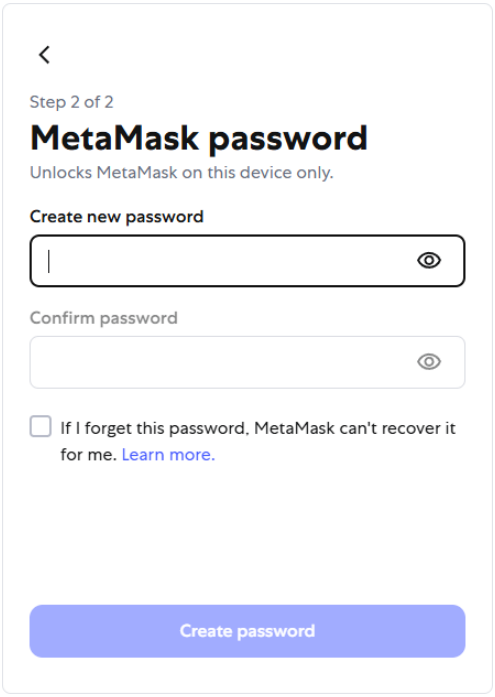
3. Import *wallet MetaMask*




**Gambar 4.3 Impor *wallet***

Gambar ini memperlihatkan proses pengimporan *wallet* di *MetaMask*. Pada layar ini, pengguna diminta memasukkan frasa pemulihan (*Seed Phrase*) dari *wallet* lama. Dengan melakukan pengimporan, pengguna dapat memulihkan akses ke *wallet* tersebut pada *MetaMask* yang baru sehingga alamat *wallet* dan saldo *token* lama tersedia untuk digunakan dalam proses distribusi *Airdrop.*

4. Selanjutnya membubat password untuk login *MetaMask*

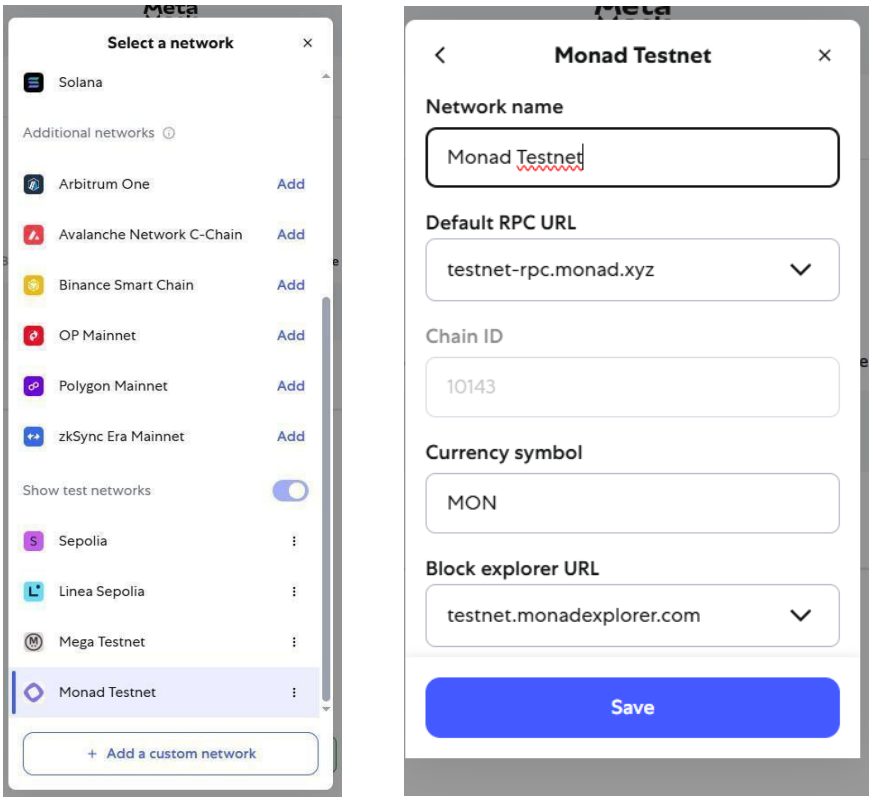



**Gambar 4.4 Membuat *password***

Gambar ini menunjukkan proses pembuatan kata sandi (*password*) untuk *wallet MetaMask*. Pengguna diminta memasukkan kata sandi baru dan mengonfirmasikannya agar dapat membuka kunci *wallet*. Tahap ini penting untuk keamanan, karena kata sandi tersebut digunakan untuk melindungi aset digital dalam *wallet* sebelum melakukan interaksi dengan jaringan *blockchain.*

5. Setelah itu, pengguna menambahkan jaringan *Monad Testnet* secara manual dengan konfigurasi:

A. Network name: *Monad Testnet*
B. *RPC* URL: https://*Testnet-RPC.Monad*.xyz
C. Chain ID: 10143
D. Currency Symbol: MON
E. Block *Explorer*: https://*Testnet.Monadexplorer*.com



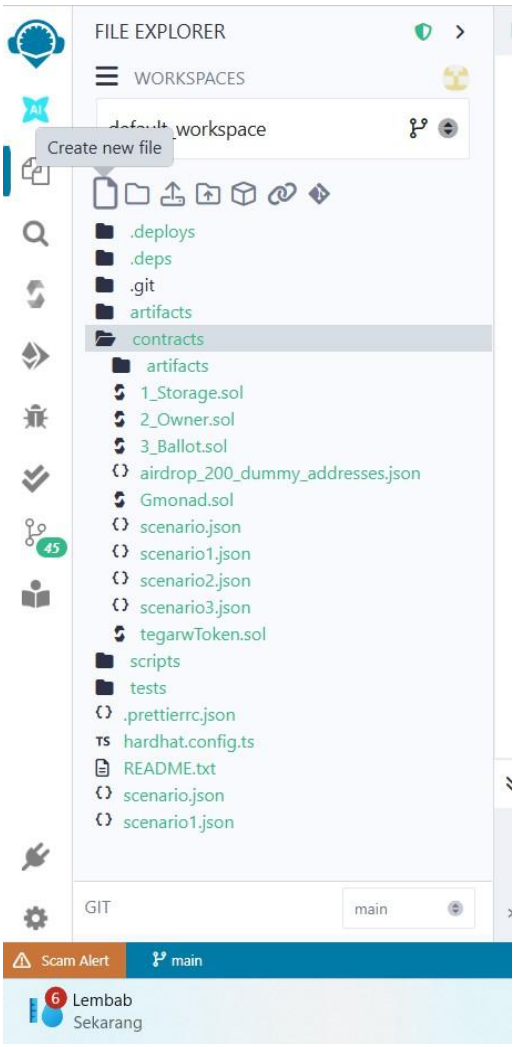

**Gambar 4.5 (a) Output konfigurasi, (b) konfigurasi *Monad***

Gambar ini menampilkan dua panel: (a) hasil konfigurasi jaringan dan (b) panel pengaturan jaringan *Monad Testnet* di *MetaMask.* Panel (b) pengguna memasukkan parameter jaringan seperti nama *network, RPC URL, Chain ID,* simbol mata uang (*MON*), dan alamat *block explorer* sesuai spesifikasi *Monad Testnet*. Setelah parameter tersebut dimasukkan, panel (a) menampilkan *output* yang mengonfirmasi bahwa jaringan *Monad Testnet* berhasil ditambahkan dan siap digunakan untuk pengujian distribusi *token Airdrop.*

### 4.1.2 Penulisan dan Kompilasi *Smart contract*

Langkah berikutnya adalah membuka *Remix IDE* https://*Remix.Ethereum*.org kemudian membuat file baru dengan nama *TegarWToken*.sol *Smart contract* ditulis menggunakan bahasa *Solidity* versi 0.8.27 dan menggunakan pustaka *OpenZeppelin* untuk memulainya sebagai berikut:

1. buat file baru menggunakan tombol "create new file" di sudut kiri atas.




**Gambar 4.6 Create new file**

Gambar ini memperlihatkan antarmuka *Remix IDE* saat pengguna membuat file baru untuk menulis *smart contract*. Pengguna mengklik tombol "*Create New File*" di sudut kiri atas lalu memberi nama file tersebut TegarW*Token*.sol. Langkah ini menyiapkan file kontrak tempat kode *Airdrop* akan ditulis menggunakan bahasa *Solidity* di lingkungan *Remix.* Setelah file dibuat, pengguna melanjutkan dengan menambahkan kode awal kontrak ke file tersebut di *editor Remix.*

2. Beri nama file baru dengan nama kode *Smart contract* yang kita buat "Tegarw*Token.sol*" dan tambahkan kode berikut ke dalamnya:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.27;
import "@OpenZeppelin/contracts/token/ERC20/ERC20.sol";
import "@OpenZeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol";
import "@OpenZeppelin/contracts/token/ERC20/extensions/ERC20Pausable.sol";
import "@OpenZeppelin/contracts/access/Ownable.sol";

contract TegarWToken is ERC20, ERC20Burnable, ERC20Pausable, Ownable {
   uint256 public constant MAX_AIRDROP_RECIPIENTS = 200;

   // Alamat owner tetap: 0x633589952c7654f78Be2412B285B04f7af37aFed
   constructor(uint256 initialSupply)
      ERC20("TegarW", "TGR")
      Ownable(0x633589952c7654f78Be2412B285B04f7af37aFed)
   {
      _mint(0x633589952c7654f78Be2412B285B04f7af37aFed, initialSupply * 10
** decimals());
   }

   // Fungsi untuk menghentikan semua transfer
   function pause() public onlyOwner {
      _pause();
   }

   // Fungsi untuk melanjutkan transfer
   function unpause() public onlyOwner {
      _unpause();
   }


 // Override untuk mendukung pausability
```

```
    function _update(address from, address to, uint256 value)
        internal
        override(ERC20, ERC20Pausable)
    {
        super._update(from, to, value);
    }

    // Fungsi Airdrop massal
    function Airdrop(address[] calldata recipients, uint256 amountPerAddress)
external onlyOwner {
        uint256 total = recipients.length;
        require(total > 0, "Recipient list is empty");
        require(total <= MAX_AIRDROP_RECIPIENTS, "Exceeds max Airdrop
limit");
        uint256 totalAmount = total * amountPerAddress;
        require(balanceOf(owner()) >= totalAmount, "Insufficient balance");
        for (uint256 i = 0; i < total; i++) {
            _transfer(owner(), recipients[i], amountPerAddress);
        }
    }
}
```

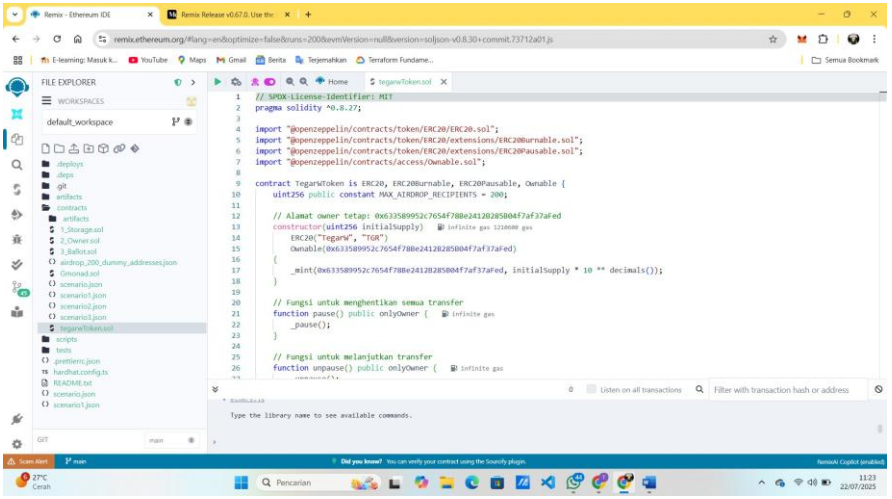
Penjelasan:

*Smart contract* TegarW*Token.sol* merupakan implementasi *token ERC*-20 berbasis *Solidity* yang dilengkapi dengan fitur *burnable, pausable*, dan distribusi *token* secara massal (*Airdrop*). Pengembangan kontrak ini menggunakan pustaka *OpenZeppelin*, yang menyediakan standar kontrak yang aman.

Kontrak ini menetapkan pemilik (*owner*) secara eksplisit dan mencetak (*mint*) sejumlah *token* awal ke alamat tersebut. Fitur *Pause* () dan *unpause* () memungkinkan pemilik menghentikan atau melanjutkan sementara aktivitas transfer, sebagai bentuk kontrol keamanan.

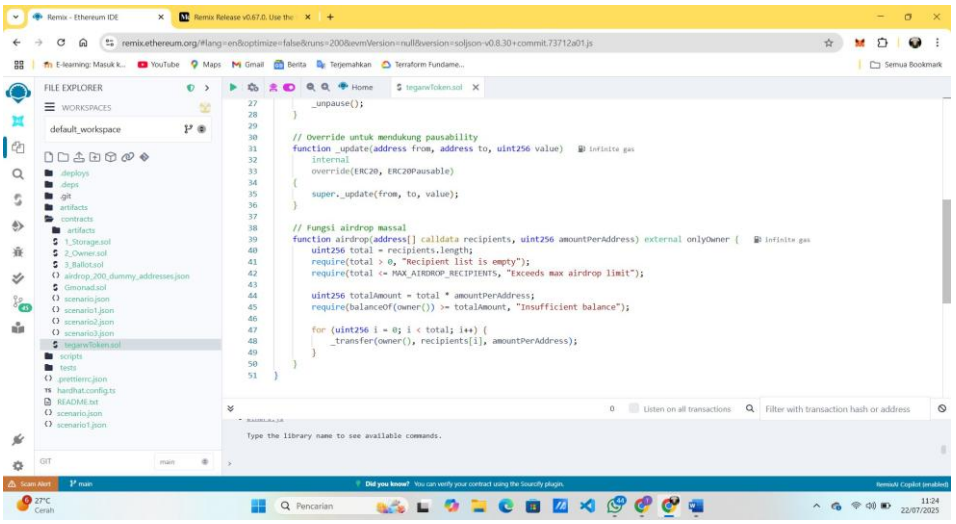
Fungsi utama *Airdrop*() dirancang untuk mendistribusikan *token* ke maksimal 200 alamat sekaligus, dengan memastikan saldo mencukupi dan hanya dapat dijalankan oleh pemilik kontrak.

Dengan struktur ini, *TegarWToken* mendukung efisiensi distribusi *token* dan kontrol akses dalam lingkungan *Blockchain Monad Testnet*.



**Gambar 4.7 Kode *Smart contract***

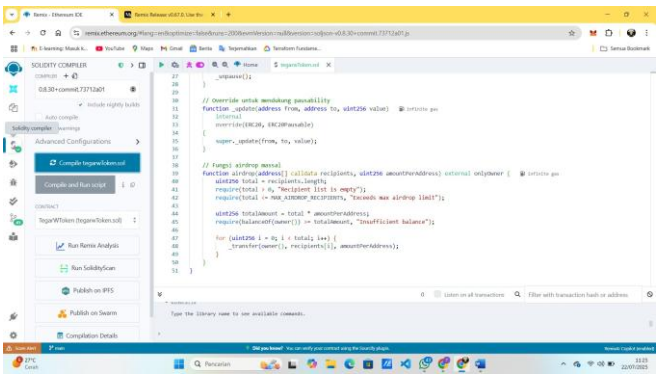
Gambar ini menampilkan potongan kode *Smart contract* berbasis *Solidity* yang dirancang untuk mendistribusikan *Airdrop*. Kode ini menggunakan pustaka *OpenZeppelin* dan mencakup fungsi utama seperti *Airdrop*, *pause*, dan unpause.



**Gambar 4.8 Kode *Smart contract***

Gambar ini menampilkan lanjutan potongan kode kontrak TegarW*Token*.sol di *Remix IDE*. Pada potongan kode ini terlihat definisi fungsi *pause(), unpause(),* dan *Airdrop*() yang khusus menambahkan logika distribusi *token* massal. Tampilan kode juga menunjukkan penggunaan pustaka *OpenZeppelin* seperti modul *ERC-20 Pausable* dan *Ownable*, yang memastikan hanya pemilik kontrak yang dapat menjalankan fungsi-fungsi penting tersebut untuk kontrol keamanan *Airdrop.*

3. Lalu *compile TegarWToken.sol*

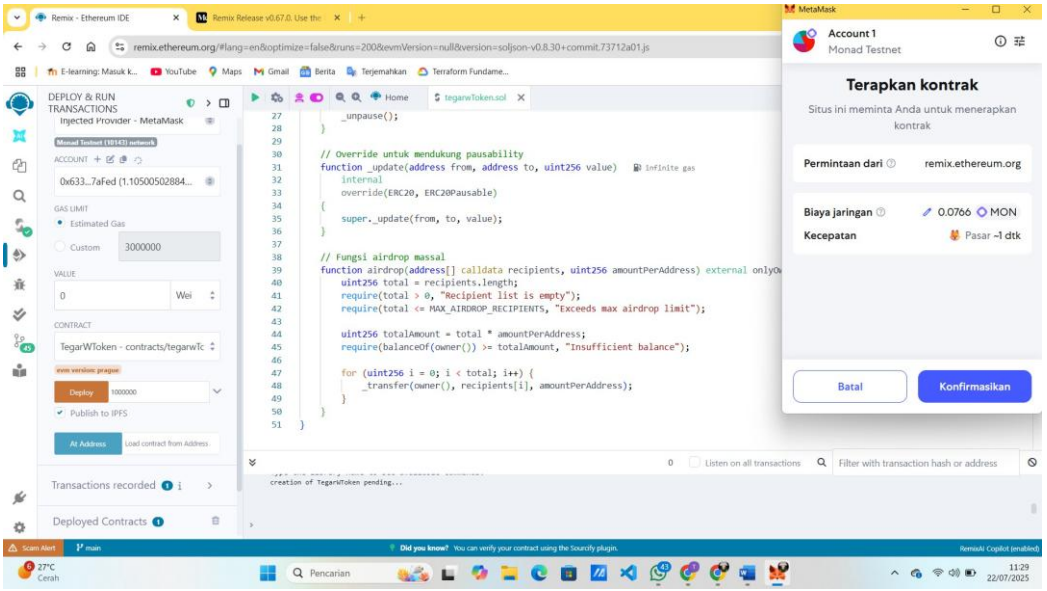


**Gambar 4.9 Compile TegarW*Token*.sol**

Gambar ini menunjukkan proses kompilasi kontrak *TegarWToken.sol* di *Remix IDE*. Setelah kode kontrak selesai ditulis, pengguna menekan tombol "*Compile*" di *Remix* untuk menghasilkan b*ytecode* kontrak. Panel *Remix IDE* menampilkan status kompilasi berhasil tanpa error, yang menandakan kontrak telah siap *dideploy* ke jaringan *Monad Testnet.*

### 4.1.3 Deploy Kontrak ke *Monad Testnet*

Setelah kompilasi berhasil, kontrak di-deploy ke jaringan *Monad Testnet* dengan memilih *environment "Injected Provider - MetaMask"* dan mengisi parameter konstruktor. *Deploy* dilakukan menggunakan akun *MetaMask* yang sudah terhubung ke jaringan *Monad Testnet* lalu klik konfirmasi untuk *deploy Smart contract*.




**Gambar 4.10 Deploy kontrak ke *Monad Testnet***

Gambar ini memperlihatkan proses *deploy smart contract* ke jaringan *Monad Testnet* menggunakan *Remix IDE* dan *MetaMask*. Pengguna memilih *environment "Injected Provider - MetaMask*" lalu memasukkan nilai awal (*initialSupply*) pada bidang konstruktor. Selanjutnya, pengguna mengonfirmasi transaksi *deploy* melalui *MetaMask*; setelah konfirmasi berhasil, kontrak *TegarWToken* akan *terdeploy* ke jaringan *Monad Testnet* dan siap dioperasikan.

### 4.1.4 Uji Coba Distribusi *Token Airdrop*

Uji coba distribusi *token* dilakukan dengan menggunakan fungsi *Airdrop* pada *Smart contract* yang telah dibuat. Pengujian dilakukan dengan 200 alamat dompet penerima dengan 5 batch, masing-masing menerima *token* yang berbeda sesuai nilai yang dimasukan. Seluruh data uji coba dijalankan di jaringan *Monad Testnet*, dengan menggunakan *batch* transfer yang mampu mengirim ke banyak alamat dalam satu kali eksekusi.

Langkah uji coba:

1. Menyiapkan *array* alamat sebanyak 200 dompet tujuan. [*dummy wallet*]

```
[ "0x0000000000000000000000000000000000000001",
 "0x0000000000000000000000000000000000000002",
 "0x0000000000000000000000000000000000000003",
 "0x0000000000000000000000000000000000000004",
 "0x0000000000000000000000000000000000000005",
 "0x0000000000000000000000000000000000000006",
 "0x0000000000000000000000000000000000000007",
 "0x0000000000000000000000000000000000000008",
 "0x0000000000000000000000000000000000000009",
 "0x000000000000000000000000000000000000000a",
 "0x000000000000000000000000000000000000000b",
 "0x000000000000000000000000000000000000000c",
 "0x000000000000000000000000000000000000000d",
 "0x000000000000000000000000000000000000000e",
```

"0x000000000000000000000000000000000000000f",
"0x0000000000000000000000000000000000000010",
"0x0000000000000000000000000000000000000011",
"0x0000000000000000000000000000000000000012",
"0x0000000000000000000000000000000000000013",
"0x0000000000000000000000000000000000000014",
"0x0000000000000000000000000000000000000015",
"0x0000000000000000000000000000000000000016",
"0x0000000000000000000000000000000000000017",
"0x0000000000000000000000000000000000000018",
"0x0000000000000000000000000000000000000019",
"0x000000000000000000000000000000000000001a",
"0x000000000000000000000000000000000000001b",
"0x000000000000000000000000000000000000001c",
"0x000000000000000000000000000000000000001d",
"0x000000000000000000000000000000000000001e",
"0x000000000000000000000000000000000000001f",
"0x0000000000000000000000000000000000000020",
"0x0000000000000000000000000000000000000021",
"0x0000000000000000000000000000000000000022",
"0x0000000000000000000000000000000000000023",
"0x0000000000000000000000000000000000000024",
"0x0000000000000000000000000000000000000025",
"0x0000000000000000000000000000000000000026",
"0x0000000000000000000000000000000000000027",
"0x0000000000000000000000000000000000000028",
"0x0000000000000000000000000000000000000029",

"0x000000000000000000000000000000000000002a",

"0x000000000000000000000000000000000000002b",

"0x000000000000000000000000000000000000002c",

"0x000000000000000000000000000000000000002d",

"0x000000000000000000000000000000000000002e",

"0x000000000000000000000000000000000000002f",

"0x0000000000000000000000000000000000000030",

"0x0000000000000000000000000000000000000031",

"0x0000000000000000000000000000000000000032",

"0x0000000000000000000000000000000000000033",

"0x0000000000000000000000000000000000000034",

"0x0000000000000000000000000000000000000035",

"0x0000000000000000000000000000000000000036",

"0x0000000000000000000000000000000000000037",

"0x0000000000000000000000000000000000000038",

"0x0000000000000000000000000000000000000039",

"0x000000000000000000000000000000000000003a",

"0x000000000000000000000000000000000000003b",

"0x000000000000000000000000000000000000003c",

"0x000000000000000000000000000000000000003d",

"0x000000000000000000000000000000000000003e",

"0x000000000000000000000000000000000000003f",

"0x0000000000000000000000000000000000000040",

"0x0000000000000000000000000000000000000041",

"0x0000000000000000000000000000000000000042",

"0x0000000000000000000000000000000000000043",

"0x0000000000000000000000000000000000000044",

"0x0000000000000000000000000000000000000045",
"0x0000000000000000000000000000000000000046",
"0x0000000000000000000000000000000000000047",
"0x0000000000000000000000000000000000000048",
"0x0000000000000000000000000000000000000049",
"0x000000000000000000000000000000000000004a",
"0x000000000000000000000000000000000000004b",
"0x000000000000000000000000000000000000004c",
"0x000000000000000000000000000000000000004d",
"0x000000000000000000000000000000000000004e",
"0x000000000000000000000000000000000000004f",
"0x0000000000000000000000000000000000000050",
"0x0000000000000000000000000000000000000051",
"0x0000000000000000000000000000000000000052",
"0x0000000000000000000000000000000000000053",
"0x0000000000000000000000000000000000000054",
"0x0000000000000000000000000000000000000055",
"0x0000000000000000000000000000000000000056",
"0x0000000000000000000000000000000000000057",
"0x0000000000000000000000000000000000000058",
"0x0000000000000000000000000000000000000059",
"0x000000000000000000000000000000000000005a",
"0x000000000000000000000000000000000000005b",
"0x000000000000000000000000000000000000005c",
"0x000000000000000000000000000000000000005d",
"0x000000000000000000000000000000000000005e",
"0x000000000000000000000000000000000000005f",

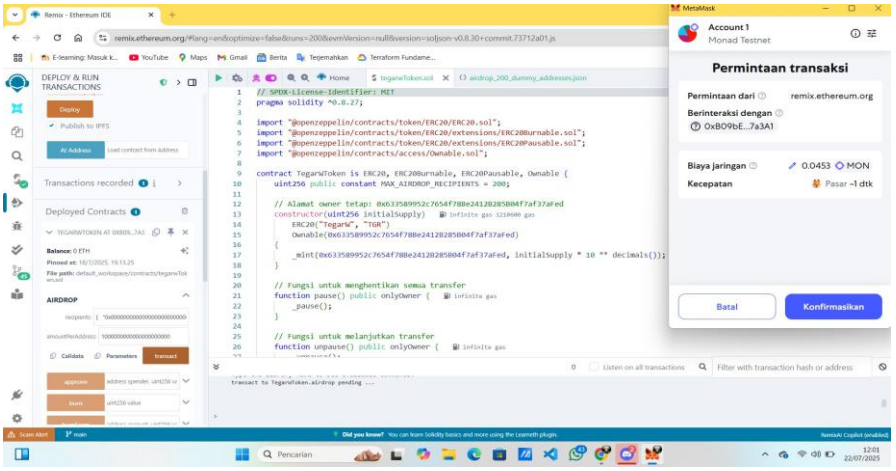
"0x0000000000000000000000000000000000000060",
"0x0000000000000000000000000000000000000061",
"0x0000000000000000000000000000000000000062",
"0x0000000000000000000000000000000000000063",
"0x0000000000000000000000000000000000000064",
"0x0000000000000000000000000000000000000065",
"0x0000000000000000000000000000000000000066",
"0x0000000000000000000000000000000000000067",
"0x0000000000000000000000000000000000000068",
"0x0000000000000000000000000000000000000069",
"0x000000000000000000000000000000000000006a",
"0x000000000000000000000000000000000000006b",
"0x000000000000000000000000000000000000006c",
"0x000000000000000000000000000000000000006d",
"0x000000000000000000000000000000000000006e",
"0x000000000000000000000000000000000000006f",
"0x0000000000000000000000000000000000000070",
"0x0000000000000000000000000000000000000071",
"0x0000000000000000000000000000000000000072",
"0x0000000000000000000000000000000000000073",
"0x0000000000000000000000000000000000000074",
"0x0000000000000000000000000000000000000075",
"0x0000000000000000000000000000000000000076",
"0x0000000000000000000000000000000000000077",
"0x0000000000000000000000000000000000000078",
"0x0000000000000000000000000000000000000079",
"0x000000000000000000000000000000000000007a",

"0x000000000000000000000000000000000000007b",
"0x000000000000000000000000000000000000007c",
"0x000000000000000000000000000000000000007d",
"0x000000000000000000000000000000000000007e",
"0x000000000000000000000000000000000000007f",
"0x0000000000000000000000000000000000000080",
"0x0000000000000000000000000000000000000081",
"0x0000000000000000000000000000000000000082",
"0x0000000000000000000000000000000000000083",
"0x0000000000000000000000000000000000000084",
"0x0000000000000000000000000000000000000085",
"0x0000000000000000000000000000000000000086",
"0x0000000000000000000000000000000000000087",
"0x0000000000000000000000000000000000000088",
"0x0000000000000000000000000000000000000089",
"0x000000000000000000000000000000000000008a",
"0x000000000000000000000000000000000000008b",
"0x000000000000000000000000000000000000008c",
"0x000000000000000000000000000000000000008d",
"0x000000000000000000000000000000000000008e",
"0x000000000000000000000000000000000000008f",
"0x0000000000000000000000000000000000000090",
"0x0000000000000000000000000000000000000091",
"0x0000000000000000000000000000000000000092",
"0x0000000000000000000000000000000000000093",
"0x0000000000000000000000000000000000000094",
"0x0000000000000000000000000000000000000095",

"0x0000000000000000000000000000000000000096",
"0x0000000000000000000000000000000000000097",
"0x0000000000000000000000000000000000000098",
"0x0000000000000000000000000000000000000099",
"0x000000000000000000000000000000000000009a",
"0x000000000000000000000000000000000000009b",
"0x000000000000000000000000000000000000009c",
"0x000000000000000000000000000000000000009d",
"0x000000000000000000000000000000000000009e",
"0x000000000000000000000000000000000000009f",
"0x00000000000000000000000000000000000000a0",
"0x00000000000000000000000000000000000000a1",
"0x00000000000000000000000000000000000000a2",
"0x00000000000000000000000000000000000000a3",
"0x00000000000000000000000000000000000000a4",
"0x00000000000000000000000000000000000000a5",
"0x00000000000000000000000000000000000000a6",
"0x00000000000000000000000000000000000000a7",
"0x00000000000000000000000000000000000000a8",
"0x00000000000000000000000000000000000000a9",
"0x00000000000000000000000000000000000000aa",
"0x00000000000000000000000000000000000000ab",
"0x00000000000000000000000000000000000000ac",
"0x00000000000000000000000000000000000000ad",
"0x00000000000000000000000000000000000000ae",
"0x00000000000000000000000000000000000000af",
"0x00000000000000000000000000000000000000b0",

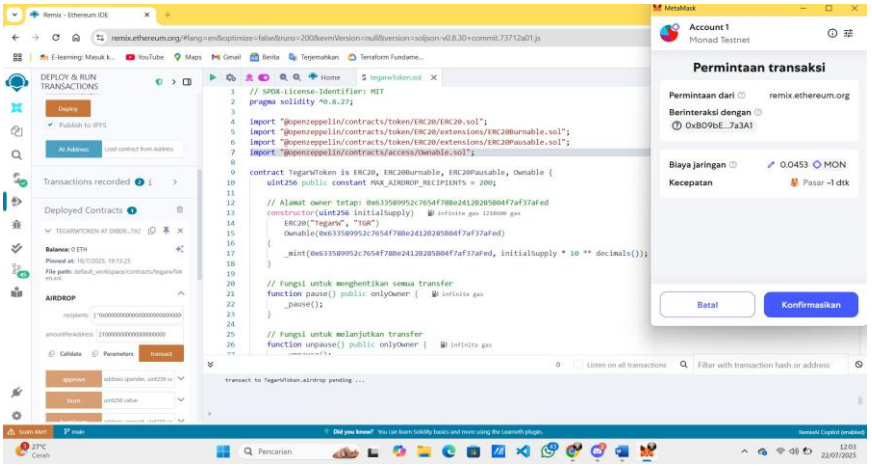
"0x00000000000000000000000000000000000000b1",
"0x00000000000000000000000000000000000000b2",
"0x00000000000000000000000000000000000000b3",
"0x00000000000000000000000000000000000000b4",
"0x00000000000000000000000000000000000000b5",
"0x00000000000000000000000000000000000000b6",
"0x00000000000000000000000000000000000000b7",
"0x00000000000000000000000000000000000000b8",
"0x00000000000000000000000000000000000000b9",
"0x00000000000000000000000000000000000000ba",
"0x00000000000000000000000000000000000000bb",
"0x00000000000000000000000000000000000000bc",
"0x00000000000000000000000000000000000000bd",
"0x00000000000000000000000000000000000000be",
"0x00000000000000000000000000000000000000bf",
"0x00000000000000000000000000000000000000c0",
"0x00000000000000000000000000000000000000c1",
"0x00000000000000000000000000000000000000c2",
"0x00000000000000000000000000000000000000c3",
"0x00000000000000000000000000000000000000c4",
"0x00000000000000000000000000000000000000c5",
"0x00000000000000000000000000000000000000c6",
"0x00000000000000000000000000000000000000c7",
"0x00000000000000000000000000000000000000c8"]

2. Menentukan jumlah *token* yang akan diberikan pada masing-masing alamat.



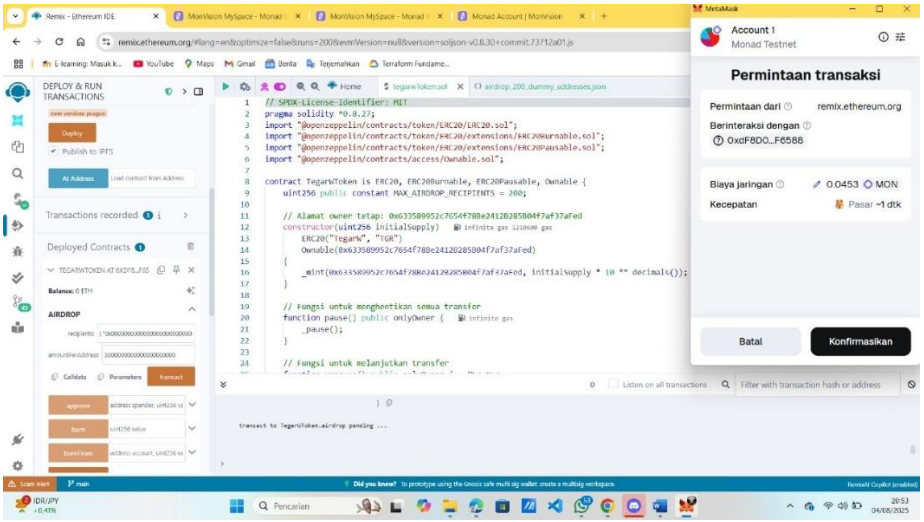

**Gambar 4.11 Menentukan jumlah *token* : 1000 TGR 100 *wallet***

Gambar ini memperlihatkan pengisian parameter fungsi *Airdrop* di *Remix IDE*, dengan jumlah 1.000 TGR *token* per penerima dan 100 alamat penerima. Pengguna memasukkan *array* alamat *wallet* serta nilai *token* per alamat ke dalam *form input.* Setelah parameter diatur, pengguna dapat mengeksekusi fungsi *Airdrop* untuk mendistribusikan *token* secara *batch* kepada 100 penerima sekaligus.



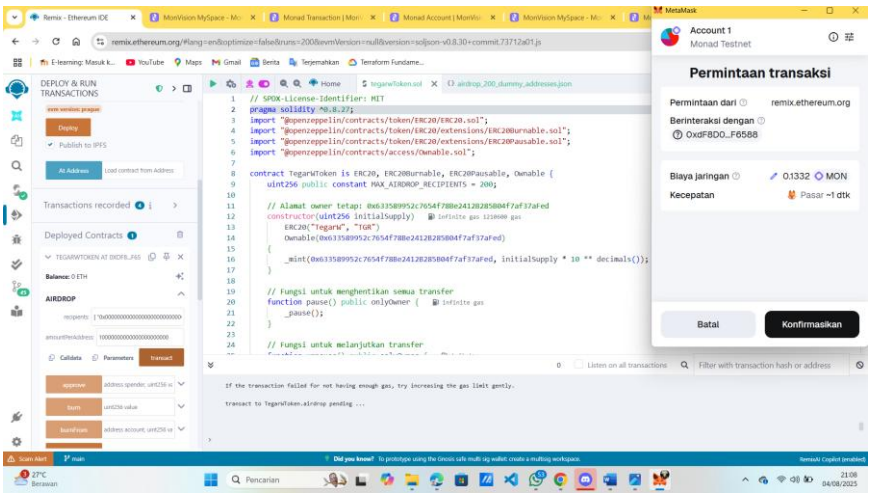

**Gambar 4.12 Menentukan jumlah *token* : 2100 TGR 100 *wallet***

Gambar ini menunjukkan langkah uji coba fungsi *Airdrop* dengan parameter baru, yaitu 2.100 TGR per penerima untuk 100 alamat. Pada antarmuka *Remix,* nilai *token* per alamat telah diubah menjadi 2.100 dan siap dieksekusi. Pengujian ini dilakukan untuk memantau penggunaan *gas fee* dan keberhasilan distribusi saat jumlah *token* per penerima ditingkatkan.



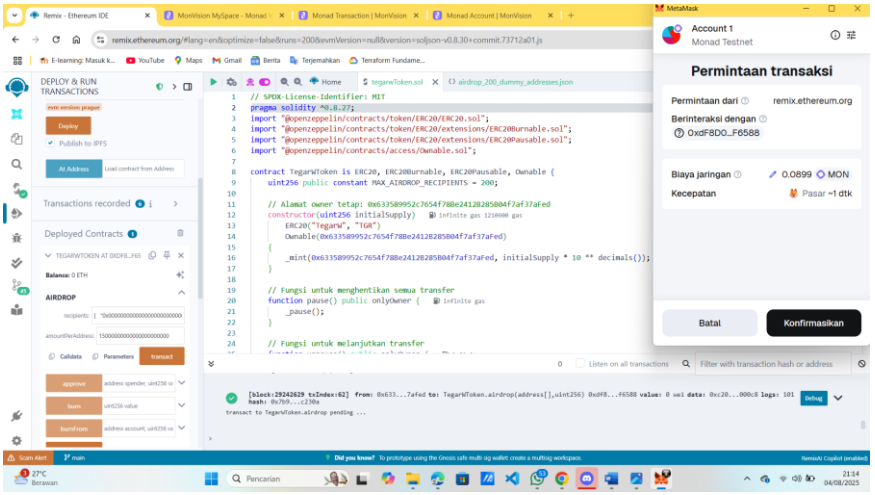

**Gambar 4.13 Menentukan jumlah *token* : 5000 TGR 100 *wallet***

Gambar ini menampilkan pengaturan fungsi *Airdrop* dengan jumlah *token* yang lebih besar, yaitu 5.000 TGR untuk setiap dari 100 penerima. Nilai *token* per alamat telah dimasukkan di form *input Remix* sesuai uji coba. Langkah ini bertujuan untuk melihat efek peningkatan jumlah *token* per penerima terhadap performa kontrak dan konsumsi *gas fee* pada jaringan *Monad Testnet.*




**Gambar 4.14 Menentukan jumlah *token* : 10.000 TGR 100 *wallet***

Gambar ini memperlihatkan pengujian distribusi dengan parameter 10.000 TGR per penerima untuk 100 penerima sekaligus. Pada antarmuka *Remix,* nilai *token* telah disesuaikan menjadi 10.000 agar dapat menguji batas atas kemampuan kontrak. Pengujian ini mengukur seberapa efisien kontrak melakukan *transfer batch* ketika beban *token* per transaksi meningkat.



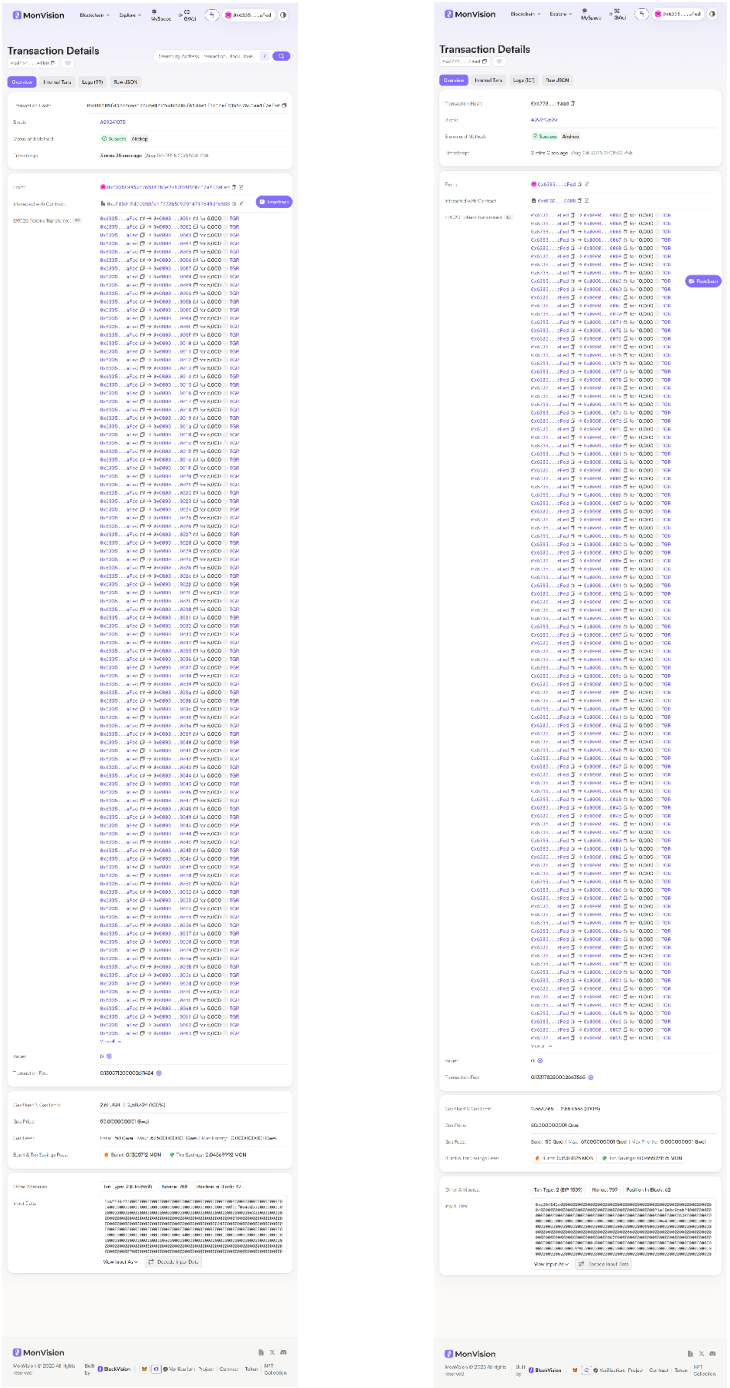

**Gambar 4.15 Menentukan jumlah *token* : 15.000 TGR 200 *wallet***

Gambar ini menampilkan pengaturan uji coba maksimal penerima, yaitu untuk 200 penerima sekaligus (sesuai konstanta MAX_AIRDROP_RECIPIENTS), Dengan 15.000 TGR per penerima. Pengguna telah memasukkan parameter tersebut pada *form input Remix* sebelum mengeksekusi fungsi *Airdrop.* Pengujian ini memastikan kontrak dapat mendukung distribusi *token* dalam jumlah besar dan memperlihatkan penggunaan *gas fee* ketika jumlah penerima maksimal sekaligus terlibat.

3. Memverifikasi hasil distribusi melalui *Monad Testnet Explorer*.

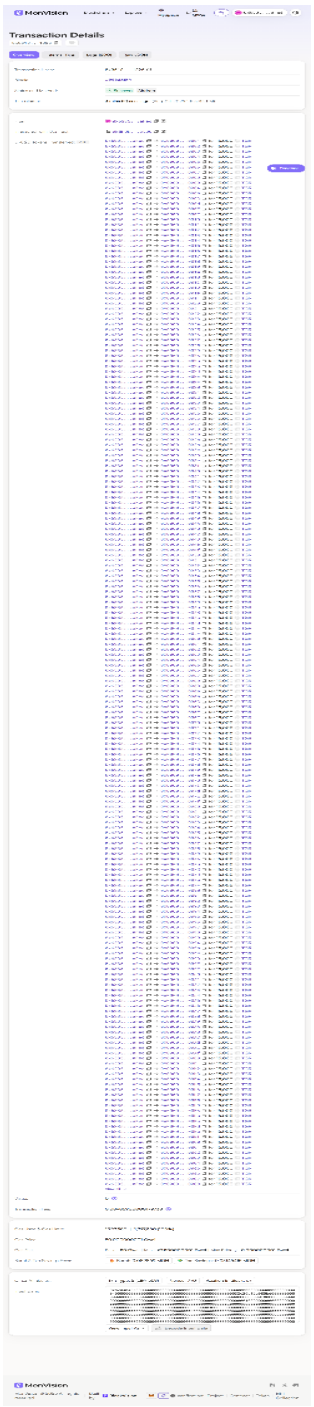

**Gambar 4.16 Hasil distribusi**

Gambar ini menampilkan ringkasan hasil distribusi *token* pada *Monad Testnet Explorer*. Tabel transaksi di *explorer* menunjukkan setiap batch distribusi *Airdrop* telah tercatat dengan status *Success,* beserta informasi jumlah *token* per transaksi dan *hash* transaksi. Informasi ini mengonfirmasi bahwa semua pengiriman *token* berhasil dieksekusi di jaringan *Monad Testnet.*




**Gambar 4.17 Hasil distribusi**

Gambar ini menampilkan ringkasan hasil distribusi *token* pada *Monad Testnet Explorer*. Tabel transaksi di *explorer* menunjukkan setiap batch distribusi *Airdrop* telah tercatat dengan status *Success,* beserta informasi jumlah *token* per transaksi dan *hash* transaksi. Informasi ini mengonfirmasi bahwa semua pengiriman *token* berhasil dieksekusi di jaringan *Monad Testnet.*




**Gambar 4.18 Hasil distribusi**

Gambar ini menampilkan ringkasan hasil distribusi *token* pada *Monad Testnet Explorer*. Tabel transaksi di *explorer* menunjukkan setiap *batch* distribusi *Airdrop* telah tercatat dengan status *Success,* beserta informasi jumlah *token* per transaksi dan *hash* transaksi. Informasi ini mengonfirmasi bahwa semua pengiriman *token* berhasil dieksekusi di jaringan *Monad Testnet.*