Programowanie systemów internetu rzeczy i aplikacji sieciowych (PSIR)

-

Projekt

Komunikacja pomiędzy IoT nodes za pomocą przestrzeni krotek

Bartłomiej Filipiuk **318762**
Kacper Usiądek **318852**
Karol Kaproń **318784**

15 stycznia 2024

Spis treści

1 Cel projektu

Celem projektu jest opracowanie API obsługującego krotki i komunikacji dzięki nim oraz opracowanie dwóch aplikacji (system rozwiązywania zadań i sprawdzanie paramatetru humidity) działających na jego podstawie.

2 Opis protkołu ALP

Opracowany protokół obłsuguje przesyłanie krotek i komunikacje dzięki nim pomiędzy wieloma modułami apliakcji. Modelowe środowisko na którym protokół został opracowany, zakłada serwer (przestrzeń krotek) oraz wiele modułów arduino komunikujących się z nim. Moduł arduino może przesyłać krotkę do przestrzeni krotek oraz odbierać ją z niej. Operacje modułów powinny zostać zaprogramowane w zależności od potrzeb aplikacji.

Komunikację rozpoczyna zawsze moduł arduino, wysyłając odpowiednie zapytanie stworzonymi w protokole funkcjami. Przesyłana wiadomość do serwera, poza samą krotką, zawiera inne istotne do odpowiedniej komunikacji informację. Takimi kluczowymi informacjami są parametry **command** oraz **tuple_name**. Dzięki takiemu podejściu, serwer poprawnie obsługuję zapytania oraz skutecznie rozróżnia poszczególne moduły będąc w stanie odesłać ewentualną odpowiedź w poprawne miejsce w całym środowisku.

2.1 Parametr command

Parametr command zawiera informację, jakie serwer ma wykonać operację. W zależności od tego parametru, aktualizuję przestrzeń krotek na dwa możliwe sposoby - dodanie krotki do przestrzeni lub usunięcie krotki z przestrzeni z odesłaniem jej do odpowiedniego modułu.

2.2 Parametr tuple_name

Parametr tuple_name zawiera nazwę krotki. W praktyce, jest to nazwa modułu od którego dostaje się aktualne zapytanie. Dzięki temu, serwer jest w stanie na nie odpowiednio zareagować i rozróżniać od kogo je dostał.

2.3 Funkcje obsługujące komunikacje z przestrzenią krotek

- **TS_OUT** - funkcja przesyłająca krotkę z modułu do przestrzeni krotek
- **TS_INP** - funkcja odbierająca krotkę z przestrzeni krotek do modułu

2.4 Przygotowanie wiadomości do komunikacji

W celu optymalnego i pomyślnego przesyłania i odbierania opisanych wiadomości, zastosowano funkcje **serializePacket** i **deserializePacket**. Są one odpowiedzalne za serializację i deserializację do pakietu za pomocą bajtów danej krotki oraz parametrów **command**, **tuple_name**, i **num_fields**.

3 Opis implementacji API

Całość API składa się z dwóch części - **tuple_space** i **udp_manager**. Tuple_space zapewnia obsługę krotek wraz ze zdefiniowanymi potrzebnymi makrami, a udp_manager odpowiada za inicjalizację komunikacji jak i samą komunikację w całym środowisku.

3.1 Opis funkcji zdefiniowanych w udp_manager

- **udp_setup** - inicjalizuję moduł ethernet arduino przydzielając losowy port z zakresu od 5000 do 6000
- **udp_send_packet** - jako argumenty przyjmuje bufor i długość pakietu. Funkcja pozwala na przesłanie wiadomości do odpowiedniego adresata (używana tuple_space)
- **udp_receive_packet** - jako argumenty przyjmuję bufor i długość pakietu. Funkcja pozwala na odbiór wiadomości (używana tuple_space)

3.2 Opis funkcji zdefiniowanych w tuple_space

- **serializePacket** - jako argumenty przyjmuję dany pakiet, command, nazwę kortki, krotkę i ilość pól w krotce. Serializuję krotkę do przesłania
- **deserializePacket** - jako argumenty przyjmuję dany pakiet, command, nazwę kortki, krotkę i ilość pól w krotce. Deserializuję krotkę do odebrania
- **initializeTuple** - jako argumenty przyjmuję daną krotkę, i dwie wartości (krotki w implementacji obsługują dwa pola)
- **ts_out** - jako argumenty przyjmuję nazwę krotki, krotkę i ilość pól. Przesyła krotkę do przestrzeni krotek
- **ts_inp** - jako argumenty przyjmuję nazwę krotki, krotkę i ilość pól. Odbiera krtokę z przestrzeni krotek

4 Opis implementacji serwera

4.1 Omówienie serwera

Serwer do działania potrzebuje swojego odpowiednika API. Zdecydowano się na taki ruch ze względu na brak w systemie dedykowanych bibliotek Zsut.

Serwer na początku inicjalizuję swoje działanie na porcie **1245**. Następnie przechodzi do głównej pętli komunikacji. Wpierw inicjalizuję on zmienne przechowujące dane klienta potrzebne do pierwszego zgłoszenia i jest gotowy na odbiór pierwszego zapytania. Po odebraniu, deserializuje je i sprawdza parametry nazwy krotki i komendy, w celu wykonania odpowiedniej sekwencji zadań. Na podstawie nazwy krotki, kopiuję adres klienta do poprzednio utworzonej zmiennej jeżeli jest potrzeba odesłania wiadomości spowrotem do odpowiedniego modułu. Po wykonaniu odpowiednich zadań, ten wypisuje aktualny stan przestrzeni krotek.

4.2 Struktury danych obsługujących krotki

Serwer obsługuje krotki na podstawie dwóch struktur.

- **krotka** - zdefiniowana w trakcie inicjalizacji sewrera, jest to struktura krotki bezpośrednio z API. W niej przechowuje krotki bezpośrednio po otrzymaniu od modułów arduino
- **przestrzeń krotek** - fizyczna przestrzeń krotek w formie listy. Tutaj przechowywane są wszystkie krotki na których podejmowane są operacje. W zależności od aplikacji może być ich wiele

5 Aplikacja 1.

5.1 Opis implementacji aplikacji

5.1.1 Moduł manager

Po inicjalizacji UDP poprzez API, moduł managera inicjalizuję krotkę zawierającą zadanie i liczbę losowaną z przedziału od 0 do 100, a następnie ją wysyła do przestrzeni krotek (do listy krotek zadań) za pomocą **ts_out**. W następnej kolejności wykonuję operację **ts_inp** w celu otrzymania krotki z wynikiem (z listy krotek wyników). Jeżeli operacja zakończy się sukcesem, wypiszę otrzymaną krotkę, odpowiednio obsłuży wynik i wypisze dostępne wyniki.

5.1.2 Moduł worker

Po inicjalizacji UDP poprzez API, moduł managera inicjalizuję odpowiednie krotki, a następnie wykonuję operację **ts_inp** w celu sprawdzenia, czy jest w przestrzeni krotek (listy zadań) zadanie do wykonania. Jeśli jest, wypiszę ją, ą następnie wykona odpowiednie operację obliczeniowe. Następnie wypisze wynik i za pomocą operacji **ts_out** wyślę ją do przestrzeni krotek (do listu krotek wyników).

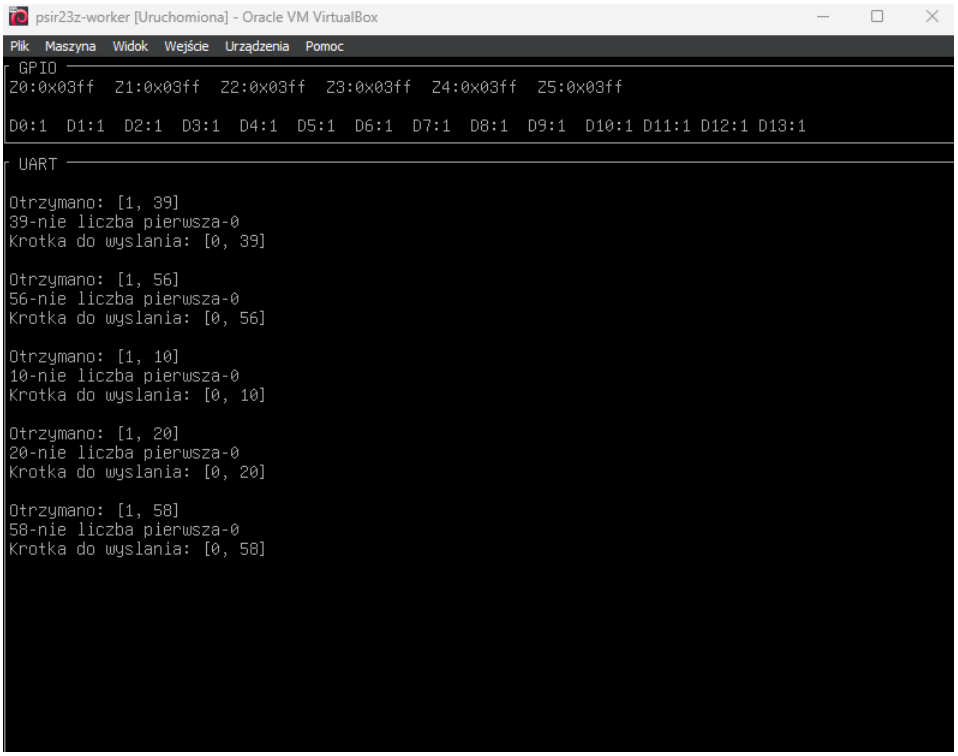
5.2 Przedstawienie działania aplikacji

```
GPIO
Z0:0x03ff  Z1:0x03ff  Z2:0x03ff  Z3:0x03ff  Z4:0x03ff  Z5:0x0001

D0:1  D1:1  D2:1  D3:1  D4:1  D5:1  D6:1  D7:1  D8:1  D9:1  D10:1 D11:1 D12:1 D13:1

UART
Otrzymano: [0, 64]
Liczby pierwsze: [73]
Liczby inne: [78,84,86,81,25,45,54,38,18,88,72,64]

Krotka wyslana pomyslnie: [1, 20]
Otrzymano: [0, 30]
Liczby pierwsze: [73]
Liczby inne: [78,84,86,81,25,45,54,38,18,88,72,64,30]

Krotka wyslana pomyslnie: [1, 19]
Otrzymano: [0, 35]
Liczby pierwsze: [73]
Liczby inne: [78,84,86,81,25,45,54,38,18,88,72,64,30,35]

Krotka wyslana pomyslnie: [1, 58]
Otrzymano: [1, 17]
Liczby pierwsze: [73,17]
Liczby inne: [78,84,86,81,25,45,54,38,18,88,72,64,30,35]

Krotka wyslana pomyslnie: [1, 74]
Otrzymano: [0, 98]
Liczby pierwsze: [73,17]
Liczby inne: [78,84,86,81,25,45,54,38,18,88,72,64,30,35,98]

Krotka wyslana pomyslnie: [1, 12]
Otrzymano: [0, 38]
Liczby pierwsze: [73,17]
Liczby inne: [78,84,86,81,25,45,54,38,18,88,72,64,30,35,98,38]

Krotka wyslana pomyslnie: [1, 65]
```

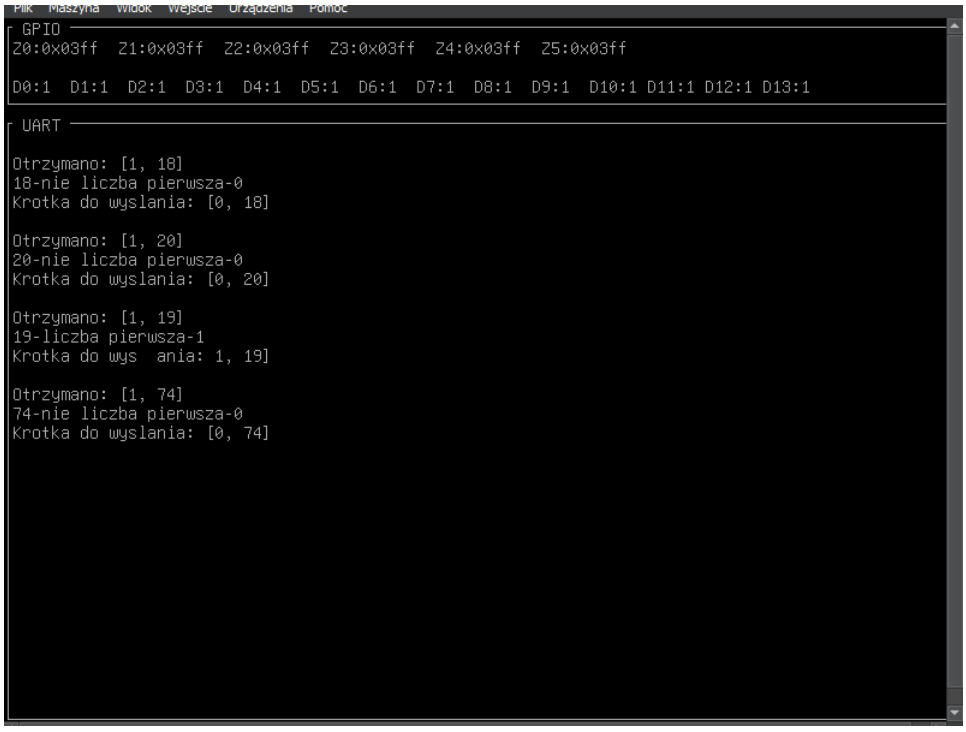
Rysunek 1: Terminal manager

```
Plik  Maszyna  Widok  Wejście  Urządzenia  Pomoc
[[1, 74], [1, 12], [1, 65]]
Przestrzeń krotek wynikow:
[[1, 7], [0, 74], [0, 46], [0, 57], [1, 5], [1, 43], [1, 79], [0, 28], [0, 72], [1, 89], [1, 53], [0
, 39], [0, 78], [0, 56], [0, 18], [0, 10], [0, 20], [0, 20], [1, 19], [0, 58]]

Otrzymano polecenie: 1, nazwa: workerReq
Wysyłanie pakietu do workera. Dane: [1, 74]
Przestrzeń krotek taskow:
[[1, 12], [1, 65]]
Przestrzeń krotek wynikow:
[[1, 7], [0, 74], [0, 46], [0, 57], [1, 5], [1, 43], [1, 79], [0, 28], [0, 72], [1, 89], [1, 53], [0
, 39], [0, 78], [0, 56], [0, 18], [0, 10], [0, 20], [0, 20], [1, 19], [0, 58]]

Otrzymano polecenie: 1, nazwa: managerReq
Wysyłanie pakietu do managera. Dane: [1, 7]
Przestrzeń krotek taskow:
[[1, 12], [1, 65]]
Przestrzeń krotek wynikow:
[[0, 74], [0, 46], [0, 57], [1, 5], [1, 43], [1, 79], [0, 28], [0, 72], [1, 89], [1, 53], [0, 39], [
0, 78], [0, 56], [0, 18], [0, 10], [0, 20], [0, 20], [1, 19], [0, 58]]

Otrzymano polecenie: 1, nazwa: workerReq
Wysyłanie pakietu do workera. Dane: [1, 12]
Przestrzeń krotek taskow:
[[1, 65]]
Przestrzeń krotek wynikow:
[[0, 74], [0, 46], [0, 57], [1, 5], [1, 43], [1, 79], [0, 28], [0, 72], [1, 89], [1, 53], [0, 39], [
0, 78], [0, 56], [0, 18], [0, 10], [0, 20], [0, 20], [1, 19], [0, 58]]

Otrzymano polecenie: 0, nazwa: workerReq
Przestrzeń krotek taskow:
[[1, 65]]
Przestrzeń krotek wynikow:
[[0, 74], [0, 46], [0, 57], [1, 5], [1, 43], [1, 79], [0, 28], [0, 72], [1, 89], [1, 53], [0, 39], [
0, 78], [0, 56], [0, 18], [0, 10], [0, 20], [0, 20], [1, 19], [0, 58], [0, 74]]
```

Rysunek 2: Terminal serwera (przestrzeń krotek)



Rysunek 3: Terminal 1. worekra



Rysunek 4: Terminal 2. workera

6 Aplikacja 2.

6.1 Opis implementacji aplikacji

6.1.1 Moduł sensing

Moduł sensing inicjalizuję UDP oraz ustawia pin 5. w tryb OUTPUT. Z tego pinu odczytywana będzie wartość 0 lub 1 parametru humidity. Parametr humidity jest symulowany za pomocą infile.txt. W trakcie głównej pętli działania, sensing czeka na zmianę wartości. Jeżeli taka następuje, wypisuję ją i za pomocą operacji **ts_out** przesyła krotkę z informacją o pinie i na jaką wartość była zmiana do przestrzeni krotek.

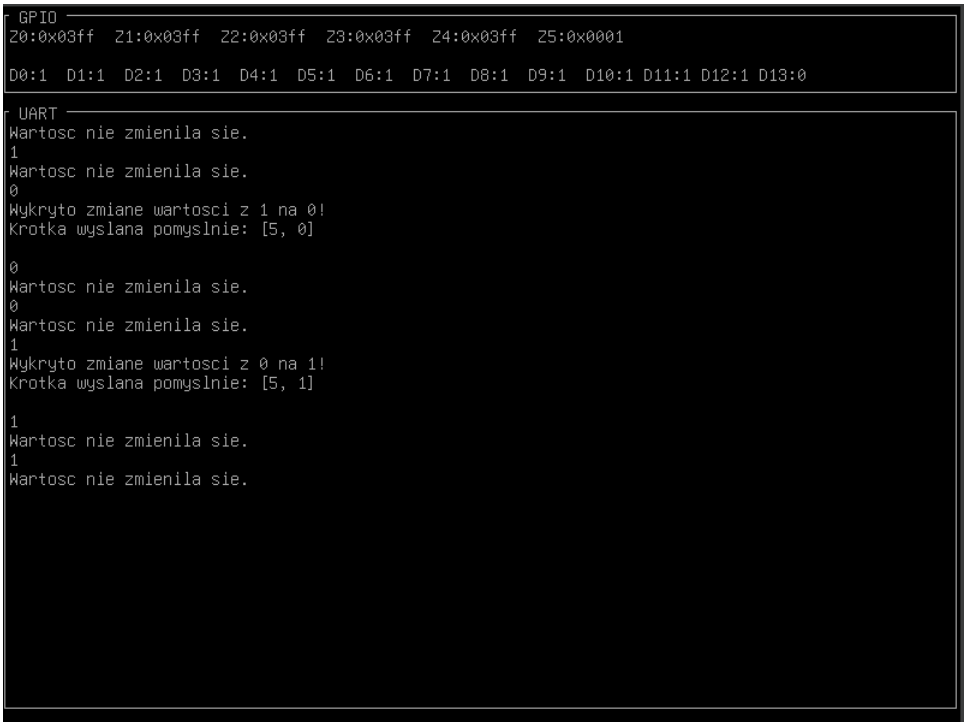
6.1.2 Moduł statsManager

Inicjalizuję UDP, a następnie przechodzi do głównej pętli komunikacji. Tam cyklicznie wykonuje operację **ts_inp** w celu zapytania, czy były jakieś zmiany parametru humidity. Jeżeli takie są w przestrzeni krotek, odbiera te dane i je wypisuję, wykonując odpowiednie operacje. Moduł ten prowadzi statystyki w postaci ilości zmian z 0 na 1 oraz z 1 na 0. Cyklicznie te informacje wypisuje.

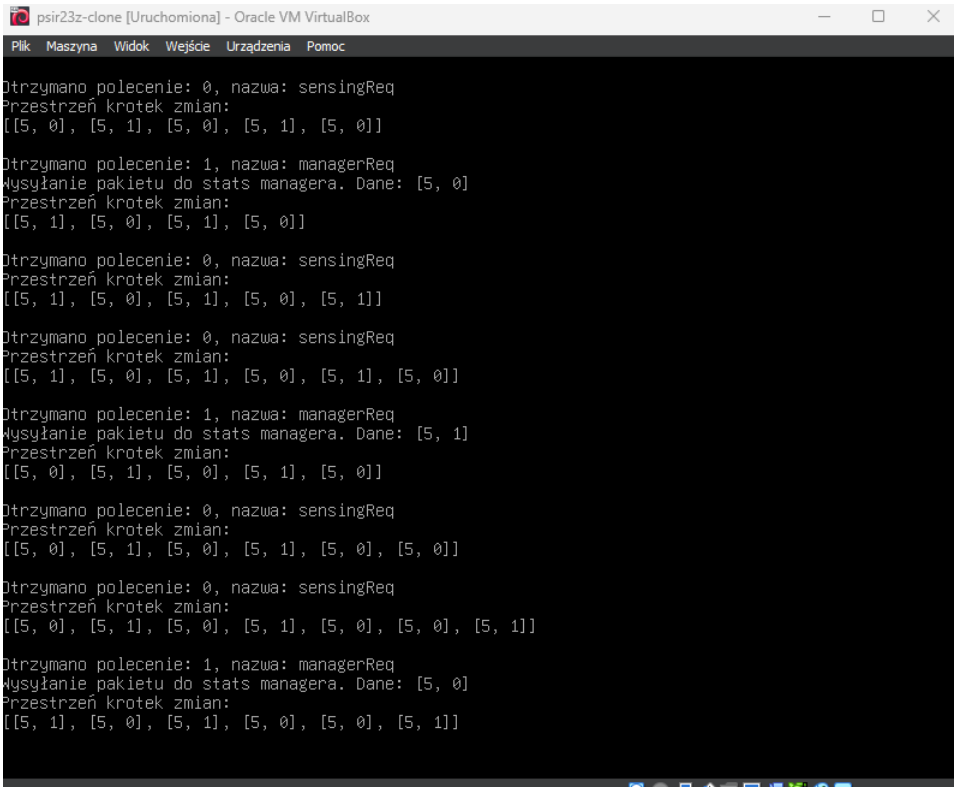
6.2 Przedstawienie działania aplikacji

```
GPIO
Z0:0x03ff  Z1:0x03ff  Z2:0x03ff  Z3:0x03ff  Z4:0x03ff  Z5:0x0000

D0:1  D1:1  D2:1  D3:1  D4:1  D5:1  D6:1  D7:1  D8:1  D9:1  D10:1 D11:1 D12:1 D13:0
UART
Wartosc nie zmienila sie.
1
Wartosc nie zmienila sie.
0
Wykryto zmiane wartosci z 1 na 0!
Krotka wyslana pomyslnie: [5, 0]

0
Wartosc nie zmienila sie.
0
Wartosc nie zmienila sie.
1
Wykryto zmiane wartosci z 0 na 1!
Krotka wyslana pomyslnie: [5, 1]

1
Wartosc nie zmienila sie.
1
Wartosc nie zmienila sie.
0
Wykryto zmiane wartosci z 1 na 0!
Krotka wyslana pomyslnie: [5, 0]

0
Wartosc nie zmienila sie.
0
Wartosc nie zmienila sie.
```

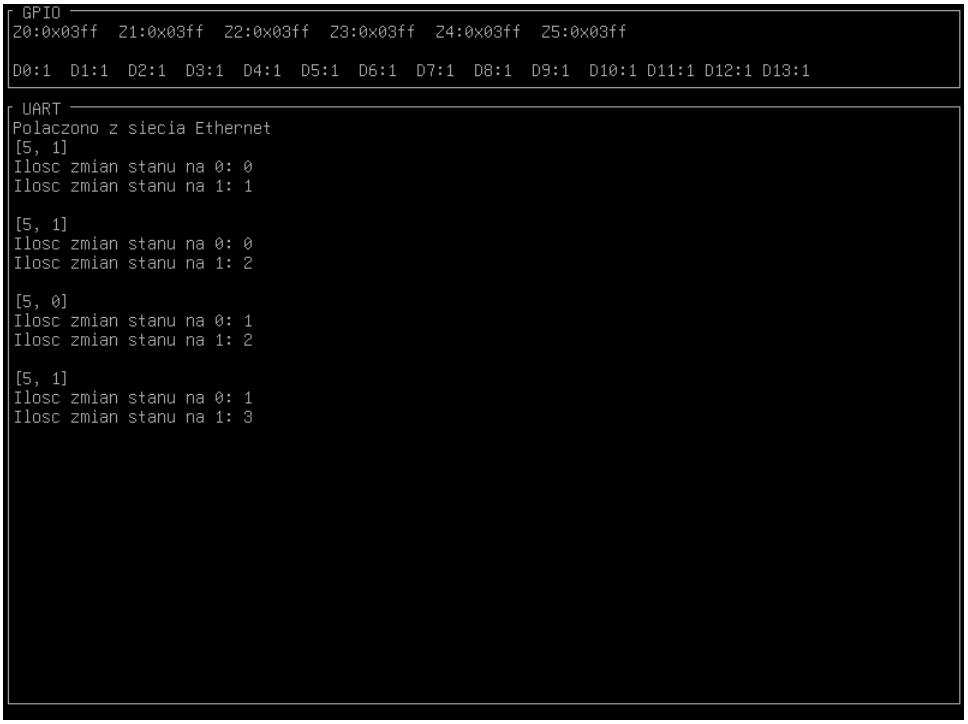
Rysunek 5: Terminal 1. sensing



Rysunek 6: Terminal 2. sensing



Rysunek 7: Terminal serwera (przestrzeń krotek)



Rysunek 8: Terminal statsManager

7 Plik symulacyjny infile.txt do Aplikacji 2.

W pliku infile przypisano różne wartości parametru humidity w zależności od czasu przebiegu symulacji. Zmiany parametru przebiegają zero-jedynkowo.