Τεχνική Αναφορά Ρομποτικού Συστήματος

1. Εισαγωγή

Η παρούσα τεχνική αναφορά περιγράφει λεπτομερώς το ρομποτικό σύστημα που έχει αναπτυχθεί για την εξερεύνηση του Άρη. Το σύστημα αποτελείται από δύο βασικές ενότητες: το αυτόνομο σύστημα και το βοηθητικό σύστημα. Σε αυτήν την αναφορά, παρουσιάζονται οι τεχνικές λεπτομέρειες του συνολικού συστήματος, συμπεριλαμβανομένων των υλικών, των ηλεκτρονικών μερών και του λογισμικού που χρησιμοποιείται.

2. Περιγραφή του Συνολικού Συστήματος

Το rover έχει σχεδιαστεί με κύριο στόχο την αναγνώριση σημαντικών πετρωμάτων στον Άρη, ενώ ταυτόχρονα πραγματοποιεί μετρήσεις περιβαλλοντικών δεδομένων, όπως η θερμοκρασία, η φωτεινότητα, η σκόνη, το CO2 και η ακτινοβολία. Οι πληροφορίες αυτές αποστέλλονται στη βάση δεδομένων στη Γη, ώστε να αξιολογηθούν από επιστήμονες.

Το σύστημα αποτελείται από:

- **Αυτόνομο Σύστημα**:Μετρούν βασικά δεδομένα του περιβάλλοντος, τα επεξεργάζονται και στα στέλνουν στη βάση.
- **Σύστημα Αναγνώρισης Πετρωμάτων**: Επιτρέπει στο rover να πλοηγείται με ελάχιστη ανθρώπινη παρέμβαση. Χρησιμοποιεί τεχνητή νοημοσύνη για την ανάλυση γεωλογικών σχηματισμών.
- **Βοηθητικό Σύστημα**:Αναπαράσταση δεδομένων περιβάλλοντος

3. Υλικά και Εξαρτήματα

Το ρομποτικό σύστημα κατασκευάστηκε χρησιμοποιώντας τα ακόλουθα υλικά και εξαρτήματα:

3.1 Υλικά Αυτόνομου Συστήματος

- **Μικροελεγκτής**: Microbit
- **Extensions:** Keystudio για τους αισθητήρες και Nezha για συλλογή πετρωμάτων.
- **Αισθητήρες**:
    - Keyestudio photoresistor light dependent resistor sensor module
    - Keyestudio DS18B20 Temperature Sensor Module
    - Keyestudio GUVA-S12SD 3528 Sunshine ultraviolet radiation sensor
    - Octopus PM2.5 Module
    - Octopus CO2 Gas Sensor
    - AILens Planetx
    - PlanetX Ultrasonic Sensor

Τεχνική Αναφορά Ρομποτικού Συστήματος

- **Κινητήρες και Μηχανισμοί Κίνησης**:
    - Μοτέρ για την κίνηση των τροχών Nezha
    - Σερβοκινητήρες για τη δαγκάνα Nezha
- **Τροφοδοσία**:
    - Μπαταρίες για την παροχή ενέργειας.

3.2 Υλικά Βοηθητικού Συστήματος

- **Μικροελεγκτής**: Microbit
    - Αισθητήρας θερμοκρασίας και ατμοσφαιρικής πίεσης.
    - Αισθητήρες φωτός και ακτινοβολίας για μέτρηση ηλιακής έκθεσης.
- **Αναπαράσταση δεδομένων**:
    - Raspberry PI και monitor

4. Λογισμικό και Προγραμματισμός

Το ρομποτικό σύστημα λειτουργεί με προγραμματισμό που περιλαμβάνει:

4.1 Προγραμματισμός Αυτόνομου Συστήματος

- **Αυτόνομη πλοήγηση**:
    - Αλγόριθμοι πλοήγησης βασισμένοι σε αισθητήρες απόστασης και τυχαίες κινήσεις.
- **Ανάλυση εικόνας**:
    - Χρήση αλγορίθμων AI για αναγνώριση πετρωμάτων.
    - Επεξεργασία εικόνας και ταξινόμηση γεωλογικών δεδομένων.
- **Επεξεργασία δεδομένων:**
    - Συλλογή δεδομένων από αισθητήρες και μετατροπή τους στην κλίμακα του 100.
    - Μέσο όρο από συνεχόμενες μετρήσεις για διασφάλιση ακρίβειας.
- **Αποστολή Δεδομένων:**
    - Επικοινωνία με ράδιο με το microbit της βάσης.
    - Κρυπτογράφηση πληροφοριών για αποφυγή παρεμβολών.

4.2 Προγραμματισμός Βοηθητικού Συστήματος

- **Λήψη Δεδομένων**
    - Το microbit λαμβάνει τις πληροφορίες και στη συνέχεια με σειριακή επικοινωνία στέλνει τα δεδομένα στο raspberry.
- **Αναπαράσταση Δεδομένων**:
    - Το raspberry λαμβάνει σειριακά τα δεδομένα από το microbit και στη συνέχεια αποθηκεύει τα δεδομένα σε ένα csv αρχείο.

Τεχνική Αναφορά Ρομποτικού Συστήματος

- Για την αναπαράσταση μία ιστοσελίδα παίρνει τα δεδομένα από το αρχείο και τα εμφανίζει σαν διάγραμμα.

5. Παράρτημα

Στο παράρτημα παρατίθεται ο κώδικας του λογισμικού που εκτελείται στα δύο υποσυστήματα.

5.1 Συλλογή, Επεξεργασία και αποστολή δεδομένων

```
from microbit import *
import radio

def avg(values):
    total = sum(values)
    count = len(values)
    average = total / count
    return average

def change(value):
    x = ((value*100)/1023)
    return round(x)

def encrypt(sensor, value):
    mes = sensor + str(value)
    return mes


def collect_send(sensor, pin):
    values= []
    for i in range(10):
        values.append(pin.read_analog())
        sleep(100)
    average = change(avg(values))
    mes= encrypt(sensor,average)
    radio.send(mes)
```

```
# Imports go at the top
from microbit import *
from data import *
import radio
radio.config(group = 2)
radio.on()
```

Τεχνική Αναφορά Ρομποτικού Συστήματος

```
pairs = {'T': pin0, 'U': pin1, 'D': pin2, 'L':pin3, 'C':pin4}
sensors = list(pairs.keys())

# Code in a 'while True:' loop repeats forever
while True:
    for i in range(len(sensors)):
        collect_send(sensors[i], pairs[sensors[i]])
```

5.2 Αυτόνομη Πλοήγηση

```
from movements import *
from microbit import *
from ai_lib import *

rock = False

def find_rock(ai):
    ai.get_image()
    if ai.get_ball_color == 'Red':
        rock = True

def rock_pos(ai, wk):
    ai.get_image()
    x, y = ai.get_ball_data()[0], ai.get_ball_data()[1]
    return (x,y)

def check_pos(ai):
    ai.get_image()
    x, y = ai.get_ball_data()[0], ai.get_ball_data()[1]
    if x < 2.5 and x > 1.5 and y < 2.5 and y > 1.5:
        return 'center'
    else:
        return 'not center'
```

```
from microbit import *
import random
from nezha import *

def choose_movement(nz, speed):
    moves = ["straight", "right", "left"]
    move = random.choice(moves)
```

Τεχνική Αναφορά Ρομποτικού Συστήματος

```
    if move == "straight":
        straight_distance(nz, speed, random.randint(1, 10))
    elif move == "right":
        right_deg(nz, speed, random.randint(1, 180))
    elif move == "left":
        left_deg(nz, speed, random.randint(1, 180))

def calc_distnce(distance, speed):
    return distance*80/100

def straight(nz, speed):
    nz.set_motors(1, speed)
    nz.set_motors(2, speed)

def straight_distance(nz, speed, distance):
    straight(nz, speed)
    sleep(calc_distnce(distance, speed))
    straight(nz, 0)

def calc_deg(deg, speed):
    return deg*10/90

def right_deg(nz, speed, deg):
    nz.set_motors(1, speed)
    nz.set_motors(2, speed / 2)
    sleep(calc_deg(deg, speed))
    nz.set_motors(1, 0)
    nz.set_motors(2, 0)

def left_deg(nz, speed, deg):
    nz.set_motors(1, speed)
    nz.set_motors(2, speed / 2)
    sleep(calc_deg(deg))
    nz.set_motors(1, 0)
    nz.set_motors(2, 0)

def init_servo(nz, servo, deg):
    nz.set_servo(servo, 90)

def rotate(nz, deg):
    nz.set_servo(1, deg)

def down(nz):
    nz.set_servo(2, 90)
```

Τεχνική Αναφορά Ρομποτικού Συστήματος

```
def up(nz):
    nz.set_servo(2, 180)

def open(nz):
    nz.set_servo(3, 90)
def close(nz):
    nz.set_servo(3, 180)

def grab_leave(nz, deg):
    rotate(nz, deg)
    down(nz)
    open(nz)
    close(nz)
    up(nz)
```

```
from microbit import *
from movements import *
from ai import *
import radio
radio.config(group = 2)
radio.on()

nz = NEZHA()
init_servo(nz, 1, 90)
init_servo(nz, 2, 90)
init_servo(nz, 3, 90)
init_servo(nz, 4, 90)

max_rocks = 4

while True:
    while max_rocks < 4:
        message = radio.receive()
        if find_rock():
            if check_pos() == 'center':
                straight_distance(nz, 100, 10)
                grab_leave(nz, 180)
        choose_movement(nz, 50)
        if message:
            if message[0] == "D":
                right(nz, 100)
                sleep(500)
                straight(0)
```

Τεχνική Αναφορά Ρομποτικού Συστήματος

5.3 Λήψη Δεδομένων

```
# Imports go at the top
from microbit import *
import radio
radio.config(group = 2)
radio.on()

sensors = ['T', 'U', 'D', 'L', 'C']

uart.init(115200)

def decrypt(message):
    return message[1:]

def warn(sensor, value): pass

while True:
    message = radio.receive()
    if message and message[0] in sensors:
        display.scroll(message)
        pos=sensors.index(message[0])
        new_message=str(pos+1)+decrypt(message)
        uart.write(new_message)
        sensor, value = message[0], decrypt(message)
```

5.4 Αναπαράσταση Δεδομένων

```
#!/usr/bin/env python
import time
import serial
from datetime import datetime
from csv import writer
import http.server

ser = serial.Serial(
        port='/dev/ttyACM0',
        baudrate = 115200,
        parity=serial.PARITY_NONE,
        stopbits=serial.STOPBITS_ONE,
        bytesize=serial.EIGHTBITS,
        timeout=1
```

Τεχνική Αναφορά Ρομποτικού Συστήματος

```
)

while True:
    data=ser.readline()
    print(data)
    if data:
        dt = datetime.now()
        datestamp = str(dt)[:16]
        newData = [datestamp,0,0,0,0,0]
        data = data.decode()
        index = int(data[0])
        newData[index]=data[1:] #data0 einai h thesh kai to data[1:] enai ta
stoixeia
        print(newData)
        with open('data.csv', 'a', newline='') as f_object:
            writer_object = writer(f_object)
            writer_object.writerow(newData)
            f_object.close()
```

```
<html>
    <head>
        <!-- Plotly.js -->
        <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
    </head>

    <body>
        <div id="comboDiv" style="width: 1000px; height: 600px;"><!-- Plotly
chart will be drawn inside this DIV --></div>

        <script>
            function makeplot() {
            Plotly.d3.csv("data.csv", function(data){ processData(data) } );

            };

            function processData(allRows) {

            console.log(allRows);
            var x = [], yTemp = [], yLight = [], yUV = [], yDust = [], yCO2
= []

            for (var i=0; i<allRows.length; i++) {
                row = allRows[i];
```

Τεχνική Αναφορά Ρομποτικού Συστήματος

```
                x.push( row['time'] );
                yTemp.push( row['temperature'] );
                yLight.push( row['light'] );
                yUV.push(row['CO2'] );
                yCO2.push(row['light']);
                yDust.push(row['UV']);


            }
            console.log( 'X',x, 'YTEMP',yTemp,
'YLIGHT',yLight,'YUV',yUV,'YDUST',yDust,'YCO2', yCO2 );
            makePlotly( x, yTemp, yLight, yUV, yDust, yCO2);
            }


            function makePlotly( x, yTemp, yLight, yUV, yDust, yCO2 ){
            //  var plotDiv = document.getElementById("plot");

            var trace1 = {
                x: x,
                y: yTemp,
                type: 'scatter',
                line: {shape: 'spline'},
                name: 'temp'
            };

            var trace2 = {
                x: x,
                y: yLight,
                type: 'scatter',
                line: {shape: 'spline'},
                name: 'light'
            };
                var trace3 = {
                x: x,
                y: yUV,
                type: 'scatter',
                line: {shape: 'spline'},
                name: 'UV'

            };
                 var trace4 = {
                x: x,
                y: yDust,
```

Τεχνική Αναφορά Ρομποτικού Συστήματος

```
                type: 'scatter',
                line: {shape: 'spline'},
                name: 'Dust'

            };
                var trace5 = {
                x: x,
                y: yCO2,
                type: 'scatter',
                line: {shape: 'spline'},
                name: 'CO2'

            };




            var traces = [trace1, trace2, trace3, trace4, trace5];

            Plotly.newPlot('comboDiv', traces,
                {title: 'micro:bit DATA'});

            };

            makeplot();

        </script>

        <button type="button" onclick="window.location.href='data.csv'"
class="primaryCTA">Download CSV file</button>

    </body>

</html>
```

6. Πηγές

Κώδικας για σειριακή επικοινωνία microbit και raspberry, και αναπαράστασης δεδομένων σε ιστοσελίδα. https://github.com/blogmywiki/microbit-pi-data