Compte rendu projet snirium :

I Partie Mobile (React Native - App.js) :

1. Interface Utilisateur (UI) :

- Mise en forme des boutons.

```
return (
  <View style={styles.page}>
    {/* Bouton pour récupérer les valeurs */}
    <View style={styles.buttonContainer}>
      <Button
        onPress={this._onPressRecuperer}
        title="Récupérer les valeurs"
        color="white"
        style={styles.button}
      />
    </View>

    {/* Boutons pour les commandes */}
    <View style={styles.arrowButtonsContainer}>
      <Button onPress={this._onPressAvancer} title="↑" color="white" style={styles.arrowButton} />
      <Button onPress={this._onPressDroite} title="→" color="white" style={styles.arrowButton} />
      <Button onPress={this._onPressGauche} title="←" color="white" style={styles.arrowButton} />
      <Button onPress={this._onPressReculer} title="↓" color="white" style={styles.arrowButton} />
    </View>

    <View style={styles.buttonContainer}>
      {/* Boutons supplémentaires */}
      <Button onPress={this._onPressTourner} title="Tourner" color="white" style={styles.arrowButton} />
      <Button onPress={this._onPressLeverBarre} title="Lever" color="white" style={styles.arrowButton} />
      <Button onPress={this._onPressStop} title="Stop" color="white" style={styles.button} />
      <Button onPress={this._onPressbaisserBarre} title="Baisser" color="white" style={styles.arrowButton} />
    </View>
```

- Affichage des valeurs récupérées du robot.

- Affichage du cercle et de la barre.

```
    </View>

    {/* Affichage des valeurs */}
    <Text>Sinirium: {this.state.donneesJson.snirium}</Text>
    <Text>Angle: {this.state.donneesJson.angle}</Text>
    <Text>Moteur Gauche: {this.state.donneesJson.moteur_gauche}</Text>
    <Text>Moteur Droite: {this.state.donneesJson.moteur_droite}</Text>
    <Text>Distance: {this.state.donneesJson.distance}</Text>

    {/* Affichage du cercle et de la barre */}
    <Svg height="200" width="200">
      <Circle
        cx="100"
        cy="100"
        r={r}
        fill={`rgb(${this.state.donneesJson.snirium * 2.5}, 0, 0)`}
        strokeWidth={5}
        stroke="white"
      />
      <Line x1={centerX} y1={centerY} x2={obstacleX+(distance/10)} y2={obstacleY+(distance/10)} strokeWidth={5} stroke="white"/>


    </Svg>
  </View>
```

2. Gestion des Commandes (Fonctions '_onPress'):

- Implémentation des fonctions '_onPressAvancer', '_onPressReculer', '_onPressTourner', '_onPressGauche',..

- Gestion des requêtes HTTP pour envoyer des commandes au robot : 'fetch'

```javascript
// Fonctions pour envoyer des commandes au robot
_onPressAvancer = () => {
  fetch('http://192.168.1.172:1665/A', {
    method: 'GET',
  });
};

_onPressReculer = () => {
  fetch('http://192.168.1.172:1665/R', {
    method: 'GET',
  });
};

_onPressTourner = () => {
  fetch('http://192.168.1.172:1665/T', {
    method: 'GET',
  });
};

_onPressGauche = () => {
  fetch('http://192.168.1.172:1665/G', {
    method: 'GET',
  });
};

_onPressDroite = () => {
  fetch('http://192.168.1.172:1665/D', {
    method: 'GET',
  });
};

_onPressLeverBarre = () => {
  fetch('http://192.168.1.172:1665/X', {
    method: 'GET',
  });
};

_onPressbaisserBarre = () => {
```

II Partie Robot (Python - main.py) :

1. Initialisation et Gestion des Capteurs/Moteurs :

- Initialisation des capteurs (couleur, ultrason, gyro) et moteurs.

```python
# Initialisation de l'EV3 Brick et des capteurs
ev3 = EV3Brick()
snirium_sensor = ColorSensor(Port.S3)
ultrasonic_sensor = UltrasonicSensor(Port.S2)
gyro_sensor = GyroSensor(Port.S4)


# Import du module
import socket


# Création des moteurs
left_m = Motor(Port.A, Direction.CLOCKWISE)
right_m = Motor(Port.C, Direction.CLOCKWISE)
medium_m = Motor(Port.B, Direction.CLOCKWISE)
```

- Création de la boucle de traitement pour gérer les commandes reçues.

- Interpréter les commandes reçues et effectuer les actions correspondantes sur le robot.

- Ajout de la logique pour récupérer les données du robot (snirium, angle, moteurs, distance) et les envoyer au client.

```
# Boucle de traitement
while True:
    # Attente de la connexion d'un client
    sdc, adresse = sds.accept()
    print("Nouveau clientE : {}".format(adresse))
    # Réception de données
    requete_octets = sdc.recv(1024)
    requete_str = requete_octets.decode("utf8")
    print(requete_str)

    # Traitement des commandes en fonction de la requête
    if requete_str[5] == "A":#Avancer
        left_m.run(500)
        right_m.run(500)
        reponse_str = "HTTP/1.1 200 OK\r\n\r\nOK"
        reponse_octets = reponse_str.encode("utf8")
        sdc.send(reponse_octets)
    elif requete_str[5] == "R":#Reculer
        left_m.run(-500)
        right_m.run(-500)
        reponse_str = "HTTP/1.1 200 OK\r\n\r\nOK"
        reponse_octets = reponse_str.encode("utf8")
        sdc.send(reponse_octets)
    elif requete_str[5] == "G":  # Tourner à gauche
        left_m.run(-500)
        right_m.run(500)
        reponse_str = "HTTP/1.1 200 OK\r\n\r\nOK"
        reponse_octets = reponse_str.encode("utf8")
        sdc.send(reponse_octets)
    elif requete_str[5] == "D":#Tourner à droite
        left_m.run(500)
        right_m.run(-500)
        reponse_str = "HTTP/1.1 200 OK\r\n\r\nOK"
        reponse_octets = reponse_str.encode("utf8")
        sdc.send(reponse_octets)
    elif requete_str[5] == "X":#Lever la barre
        medium_m.run(-1000)
        wait(1000)
        medium_m.stop()
```

2. Serveur TCP et Gestion des Connexions :

- Mise en place du serveur TCP pour écouter les connexions des clients.

- Gestion de l'acceptation de nouvelles connexions et de la réception des requêtes.

```
# Création de la socket TCP
sds = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sds.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,1)
# Attachement de la socket
sds.bind(("0.0.0.0", 1665))

# Création d'une file d'attente pour les clients
sds.listen(5)
```

```python
while True:
    # Attente de la connexion d'un client
    sdc, adresse = sds.accept()
    print("Nouveau clientE : {}".format(adresse))
    # Réception de données
    requete_octets = sdc.recv(1024)
    requete_str = requete_octets.decode("utf8")
    print(requete_str)
```

```python
  reponse_str = http_header + http_payload
  reponse_octets = reponse_str.encode("utf8")
  print(reponse_octets)
  sdc.send(reponse_octets)

se:
  print("???")
```

III The final result :