



Unit 5: Rover センサ

Skill Builder 1: Ranger 紹介

このレッスンでは、TI-Innovator Roverの正面から障害物までの距離を読み取る方法を学習します。その距離を監視し、2つの異なる方法で情報を表示します。もう1つのセンサはRoverの下部にあり、色を「見る」ことができます。

目標

- RoverのRanger測定値を読む
- **Print()**を使って画面表示
- **text_at()**を使って画面表示
- 測定単位の決定

Rover前面にある2つの小さなシリンダーは、ヘッドライトではありません。超音波距離センサです。サイレント(私たちにとって)トーンが2つのセンサの一方から送信され、もう一方は音が障害物に当たって跳ね返ったときにエコーを聞き取ります。つぎに、内部ソフトウェアは音速と、エコーがセンサに戻るのにかかる時間を使って障害物までの距離を計算します。

$$D = V * T$$

それはすべてあっという間に行われます。

1. 新規のPython Rover Codingプロジェクトを開始します。このレッスンでは、便利な「escを押して終了」ループを使います。

```
while get_key() != "esc":
    block
```

このコマンドはメニューの数か所にありますが、**Rover**コマンドを使っているため **menu > TI Rover > Commands**(メニュー>TI Rover>コマンド)を参照してください。

2. **while**ブロックに次の3つのステートメントを追加します。

- a) Roverの前の距離を読みます。

```
dist = rv.ranger_measurement()
```

(これは、**menu > TI Rover > Inputs** (メニュー>TI Rover>入力)にあります。)

- b) 画面に印刷(表示)します。

```
print("Distance= ",dist)
```

- c) 次の距離を読み取る前に待機します。

```
sleep(.5)
```

プログラムを実行し、Roverの正面で手を近づけたり遠ざけたりします。Roverを壁、天井、床に向けます。表示された値に注意します。

Note: Roverは動きません...まだ。

使われている距離の単位(フィート、メートルなど)を指定できますか。



```
1.1 1.2 1.3 *Unit5 Pyt_ors RAD 10/22
*u5sb1.py
# Rover Coding Unit 5 SB1
#=====
import ti_rover as rv
from math import *
import ti_plotlib as plt
from ti_system import *
from time import *
#=====
while get_key() != "esc":
    block
```

```
1.1 1.2 1.3 *Unit5 Pyt_ors RAD 13/24
*u5sb1.py
from time import *
#=====
while get_key() != "esc":
    dist=rv.ranger_measurement()
    print("Distance= ",dist)
    sleep(.5)
```



3. 印刷された数値を画面下にスクロールするのではなく、**tiplotlib**モジュールにある**text_at()**関数を使うことで距離の表示を改善します。前のTI-Innovator™ Hubのレッスン(ユニット1, 2, 3)をやったことがあるなら、そのコマンドを使っています。ただしここでは、Roverテンプレートにインポートされた方法のため、モジュールの(一時的な)名前を指定する必要があります。

import tiplotlib as plt

これには、**tiplotlib**モジュールにあるすべての関数の前に、別名**plt**を付ける必要があります。また、描画画面をクリアするモジュールにある**cls()**関数を使います。

```

# Rover Coding Unit 5 SB1
#=====
import ti_rover as rv
from math import *
import tiplotlib as plt
from ti_system import *
from time import *
#=====
while get_key() != "esc":
    dist=rv.ranger_measurement()
    print("Distance= ",dist)

```

4. 次の関数を見ましょう。

plt.cls() plt.text_at(...)

これらは、**menu > TI PlotLib > Draw**(メニュー>TI PlotLib>描画)にあります。

このモジュールにある他の関数はグラフ化に使われ、次のユニットではRover座標を使うときに役立ちます。

5. **print()**ステートメントで**ctrl+T**を押して、**print()**ステートメントを**#comment**に変換します。

2つの**plt**関数を追加します。

plt.cls() plt.text_at(7, str(dist), "left")

これらは、**menu > TI PlotLib > Draw**(メニュー>TI PlotLib>描画)にあります。

```

#=====
while get_key() != "esc":
    dist=rv.ranger_measurement()
    # print("Distance= ",dist)
    plt.cls()
    plt.text_at(7,"dist","left")
    sleep(.5)

```

再び、**str(dist)**について：**plt.text_at(row, "text", "align")**関数は、引用符で囲まれたリテラル文字列(文字そのもの)、または文字列を含む変数のテキストのみを表示できます。**dist**は数値を含む変数です。そのため、次のところにある**str()**関数を使って文字列に変換する必要があります。

menu > Built Ins > Type(メニュー>組み込み>タイプ)

row(行)7は、この画面の13行の垂直中心線です。

小さい**sleep()**値を使うと、距離をより速くサンプリングできます。

これで、デジタル巻尺が作成されました。

次のレッスンでは、このことを使ってRoverが何かに衝突するのを防ぎます。