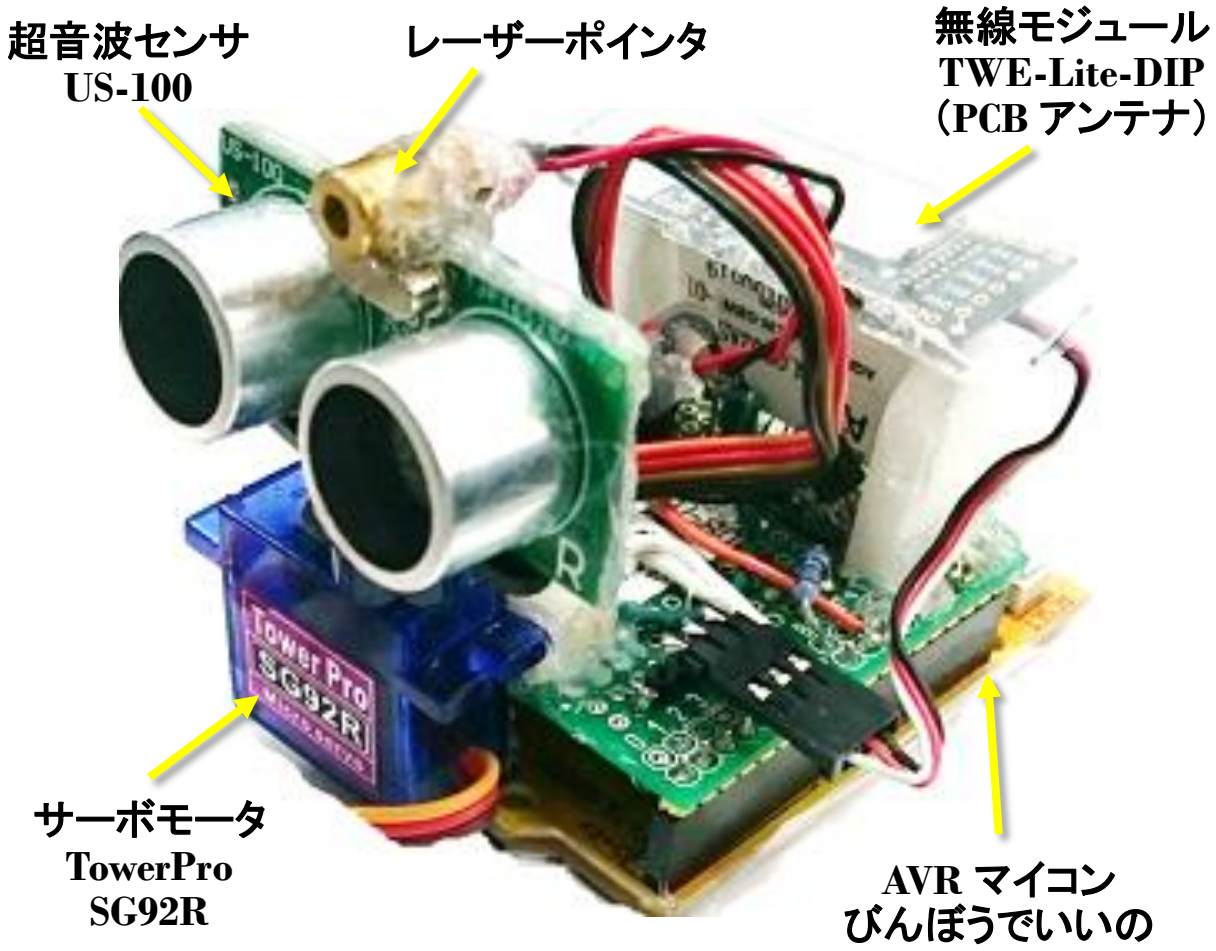
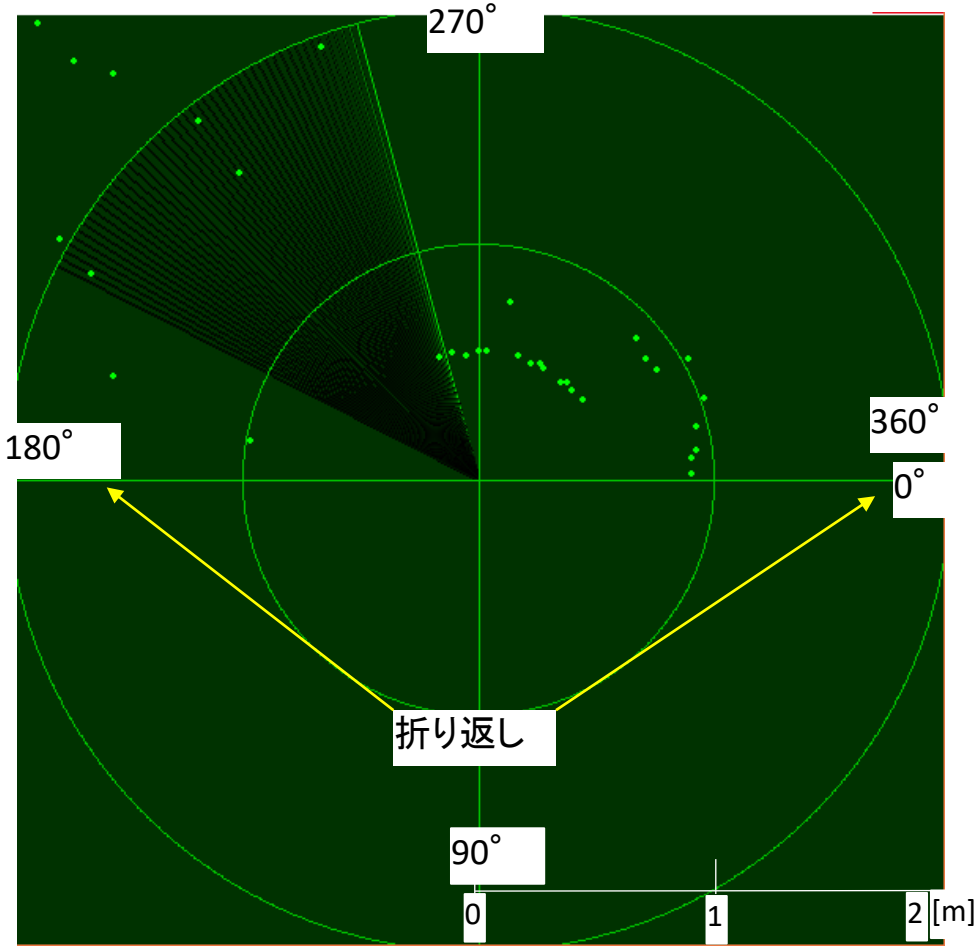
センサ ワイプ方式を用いたワイヤレス　リアルタイムレーダ

TWE-Lite-DIP(MONO WIRELESS : ZigBee)による無線化





Python (pygame)による描画

TWE-Lite-USB
受信用（送信可）

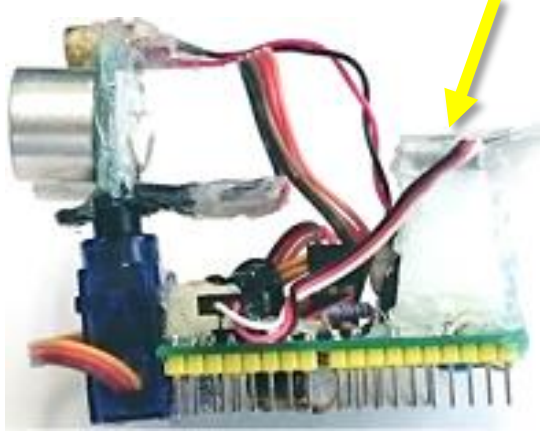
http://mono-wireless.com/jp/products/TWE-Lite-DIP/selection.html

TWE-Lite-DIP　の透過モード設定

http://www.geocities.jp/mtakapii/TWE_Lite_Set.pdf

無線モジュール（TWE-Lite-DIP）などの部品取り付け

TWE-Lite-DIP



差し込む

AVR マイコン
びんぼうでいいの

レーザーポインタ

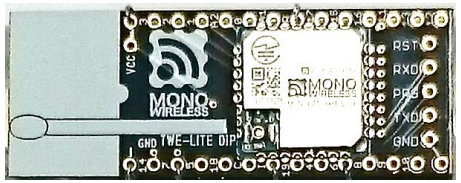
無線モジュール
TWE-Lite-DIP （UART）

配線数 ： Vcc, GND, RXD 合計3本
制御指令をマイコンに送る時はTXDを追加

LCD AQM0802A
距離、速度、角度
加速度、角度等表示

グルーガン

接着物
- 無線モジュール、
- サーボモータ、
- マイコン裏面絶縁シート
- LCD，その他



TWE-Lite-DIP

http://mono-wireless.com/jp/products/TWE-Lite-DIP/selection.html

受信データの前処理

受信データは，様々な雑音（障害）によって，右のように不揃いで，距離や角度が異常な値が含まれる．
このようなデータをそのまま処理すると，python が停止して都合が悪い．そこで，次のように予め異常データを検査して、データ処理するか無視するかを判断する．

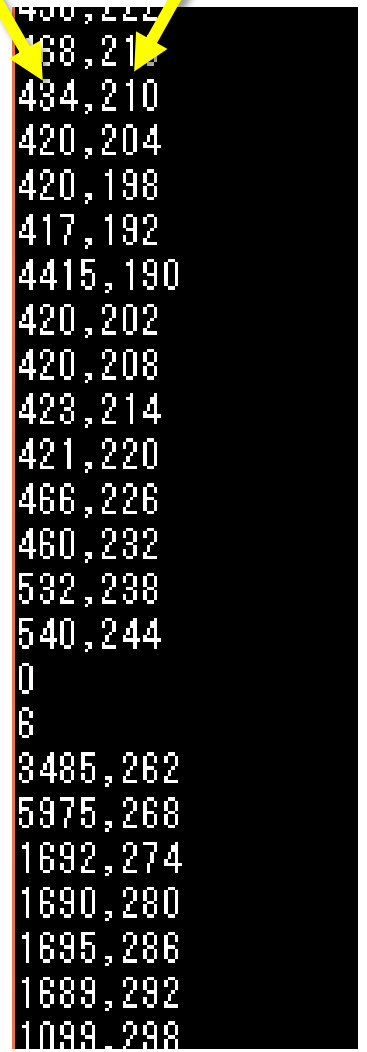
処理する条件：
- “，”が1個の場合

無視する条件：
- 文字列が何も無い場合（null）
- 距離データが4000[mm]以上の場合
- 角度が180[degree]未満，または361[degree]以上の場合

以上の条件が全て、揃った時，次の処理に進む．
これにより，ほぼ停止することなく安定に描画できる．

データの品質がある程度保証されるパケット通信などと異なり，使う側に，データの曖昧さを考慮したプログラミングの必要性が要求される．これは，このデバイスをこのモード（透過）で使用する宿命と言える．

距離 [mm]　角度 [degree]



サンプル

Arduino マイコン側 （このシリーズの首振りなしの場合と基本的に同じ）

```
/* ********************************************
 * 2016.10.22  mtsw  US-100
 * US-100 & AQM0802A == Distance + angle
 * Non Timer
 USART ... Universal Shynchrounous Asyncrhrounous
        Receiver Transmitter
        汎用同期 非同期 送受信
 ********************************************
 * This code is tested by iseerobot.com
 * -QC iseerobot -
 *   https://github.com/iseerobot/US-100-Y401
 */
// Wiring :
// VCC = 5V  // GND = GND
// Trigger = PIN digital 6
// Echo    = PIN digital 7
// NOTE: don't forget to pull jumper on US-100

#include <Wire.h>
#include <ST7032.h>  //AQM0802A control chip
#include <Servo.h>
#define AQM0802A
#ifdef AQM0802A
ST7032 lcd;
#endif

const int LASER   = 8;
//const int trigger = 6;
const int trigger = 4;
const int echo    = 7;
const int servono = 9;
const int step    = 6;
const float angle = 0;
volatile float distance;
volatile static boolean output = HIGH;

Servo servo;

void setup() {
#ifdef AQM0802A
  lcd.begin(8, 2);
  lcd.setContrast(30);
  lcd.print("Taka Lab");
  lcd.setCursor(0, 1);
  lcd.print("US-100");
#endif
  Serial.begin(115200);
  //Serial.begin(9600);
  servo.attach(9);
  //Serial.println();
  //Serial.println("I'm Mitsuo Takahashi 2016.9.1");
  //Serial.println("US-100 Sonic Sensor ==> AQM0802A＋
USART");
  //Serial.println("Measurment Distance & Velocity");
  pinMode(trigger, OUTPUT);   // Super Sonic Sensor
  pinMode(echo, INPUT);       // Super Sonic Sensor
  pinMode(LASER, OUTPUT);     // Laser Pointer
}

float Meas() {              // dimension ... [mm]
  float data;

  data = 0;
  digitalWrite(trigger, LOW);
  delayMicroseconds(5);
  digitalWrite(trigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger, LOW);
  distance = pulseIn(echo, HIGH);
  if (distance > 0) {
    distance /= 2;
    distance = distance * 340 * 100 / 100000;
    data += distance;
  }
}

void loop() {
  float deg  = 0;
  const float stangle =  10;
  const float offset  =  10;
  const int servorate =  12;
  // const float enangle = 175;
  // 0-180
  for ( deg = stangle; deg < 180; deg ) {
    servo.write(180 - deg + offset);
    Meas();
    delay(servorate);
    Serial.print(int(distance));
    Serial.print(",");
    Serial.println(int(deg + 180));
    deg += step;
#ifdef AQM0802A
    lcd.setCursor(0, 0);
    lcd.print("       ");
    lcd.setCursor(0, 0);
    lcd.print(int(distance));
    lcd.print(" mm");
    /*
      lcd.setCursor(0, 1);
      lcd.print("       ");
      lcd.setCursor(0, 1);
      lcd.print(Velo1);
      lcd.print(" km");
      */
#endif
    digitalWrite(LASER, output);
    output = !output;
   }
   // 360-180
  for ( deg = 180; deg > stangle; deg ) {
    deg -= step;
    servo.write(180 - deg + offset);
    Meas();
    delay(servorate);
    Serial.print(int(distance));
    Serial.print(",");
    Serial.println(int(deg + 180));
#ifdef AQM0802A
    lcd.setCursor(0, 0);
    lcd.print("       ");
    lcd.setCursor(0, 0);
    lcd.print(int(distance));
    lcd.print(" mm");
    /*
      lcd.setCursor(0, 1);
      lcd.print("       ");
      lcd.setCursor(0, 1);
      lcd.print(Velo1);
      lcd.print(" km");
      */
#endif
    digitalWrite(LASER, output);
    output = !output;
   }
}
```

Python (pygame) 側

```
# -*- coding: utf-8 -*-
# radar: RAdio Detecting And Ranging
# http://denshi.blog.jp/arduino/python_pygame_radar
# （ほとんどそのまま動作しました．感謝致します）
# sweep 速度は通信速度，センサの測定速度．
# 制御マイコンの処理速度，音速などに依存します，
# 実験では遠距離ほど遅い rate となります．
# 音波伝搬速度　≒ 331.5＋0.607×t [m/s]　tは摂氏温度
# 電磁波伝搬速度≒ 3*10^8 [m/s]
# 無資格運用が可能な魚群探知レーダーは約4kWなので、
# レーダーの近傍（船底）では注意が必要でしょう！
# pyserial をInstall し、Arduino からの不要文字列出力を
# 停止すると、このソースがそのまま動作しました
#
# 2016.10.15 Fri. mtakapii

import sys
import pygame
import numpy as np
from pygame.locals import *  # Python ではfromとpygameの間に2個以上の
                             # スペースは不可．mtaka
import serial

def main():

    (w,h) = (600,600)  # 画面サイズ
    deg = 0            # 初期角度
    x = [0]*50         # 障害物のx座標
    y = [0]*50         # 障害物のy座標
    pygame.init()      # pygame初期化
    # pygame.display.set_mode((w, h), 0, 32)  # 画面設定
    pygame.display.set_mode((w, h), 1, 32)  # 画面設定

    pygame.display.set_caption("Radar")
    screen = pygame.display.get_surface()
    ser = serial.Serial("COM9",115200)  # COMポート(Arduino接続)

    scale1 = 6.64      # grid scale 1m
    scale  = scale1*2  # grids cale 2m

    i = 0
    while i <= 3 :    # ゴミを含んだ初期データを捨てる．mtaka
        ser.readline()
        i += 1

    while (1):
        data = ser.readline().rstrip()  # ¥nまで読み込む(¥nは削除)
        n = data.count(',')
        if n == 1 :
            (L, deg) = data.split(",")            # mtaka
            if L != '' and deg != '':             # Not null
                m = float(L)
                n = float(deg)
                if  m>1 and m<4000  and n>1 and n < 361:
                    (L, deg) = (int(L)/scale, int(deg))    # mtaka
                # レーダービームの軌跡描画
                    for i in range(1, 50):
                        dx = w/2 * np.cos(np.radians(deg-i)) + w/2
                        dy = h/2 * np.sin(np.radians(deg-i)) + h/2
                        pygame.draw.aaline(screen, (0, 235/i+20, 0), (w/2, h/2), (dx, dy),0)
        # レーダー画面の目盛描画
        # pygame.draw.circle(screen, (0, 200, 0), (w/2, h/2), w/2, 1)
                pygame.draw.circle(screen, (0, 200, 0), (w/2, h/2), w/2, 1)
                pygame.draw.circle(screen, (0, 200, 0), (w/2, h/2), w/4, 1)
                pygame.draw.line(screen, (0, 200, 0), (0, h/2), (w, h/2))
                pygame.draw.line(screen, (0, 200, 0), (w/2, 0), (w/2, h))
        # 障害物の描画
                x0 = int(L*np.cos(np.radians(deg))) + w/2
                y0 = int(L*np.sin(np.radians(deg))) + h/2
                x.pop(49)
                y.pop(49)
                x.insert(0,x0)
                y.insert(0,y0)
                for i in range(1, len(x)):
            # pygame.draw.circle(screen, (0, 255, 0), (x[i], y[i]), 3)
                    pygame.draw.circle(screen, (0, 255, 2), (x[i], y[i]), 2)

                pygame.display.update()       # 画面更新
        # screen.fill((0, 20, 0, 0))    # 画面の背景色
                screen.fill((0, 50, 0, 0))    # 画面の背景色

        # イベント
                for event in pygame.event.get():
                    if event.type == QUIT:    # 閉じるボタンが押されたら終了
                        pygame.quit()         # Pygameの終了(画面閉じられる)
                        sys.exit()

if __name__ == "__main__":
    main()
```