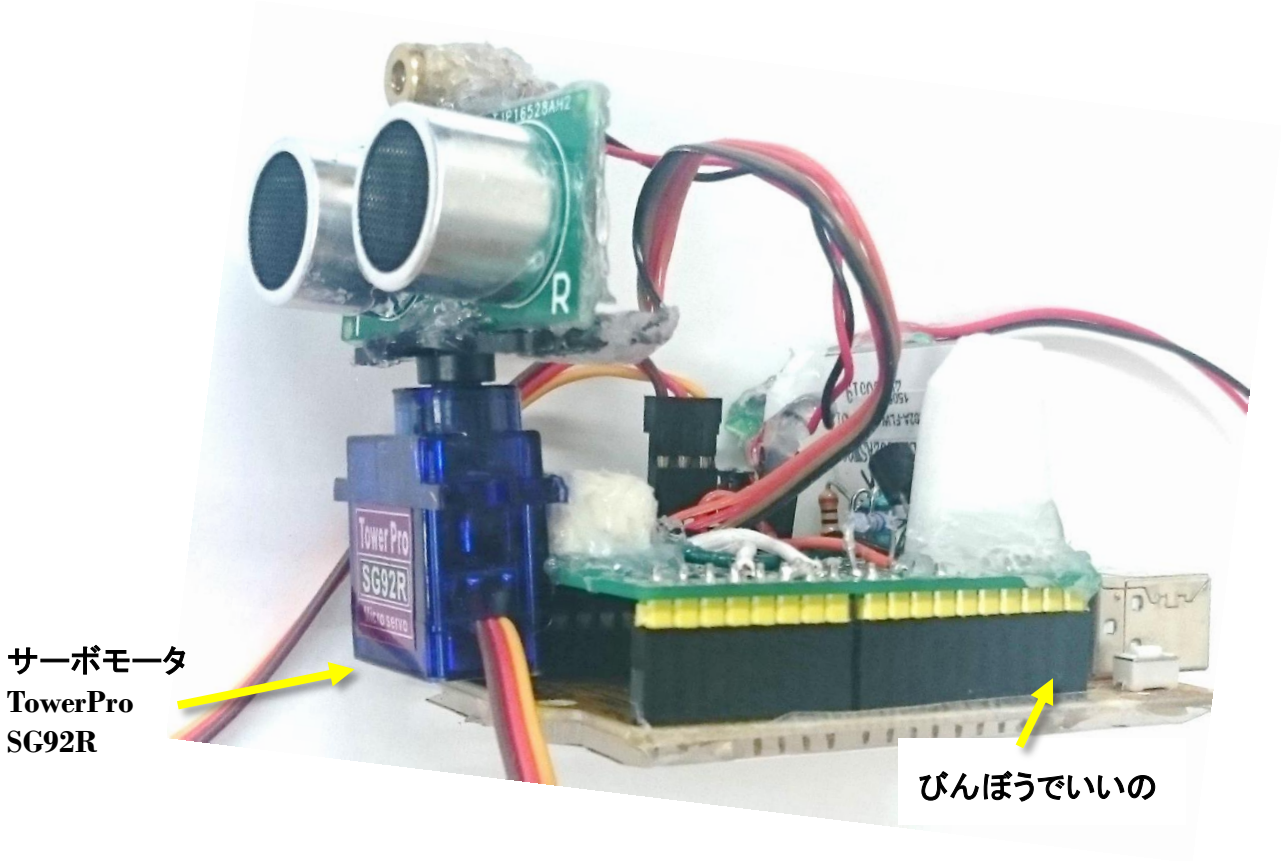
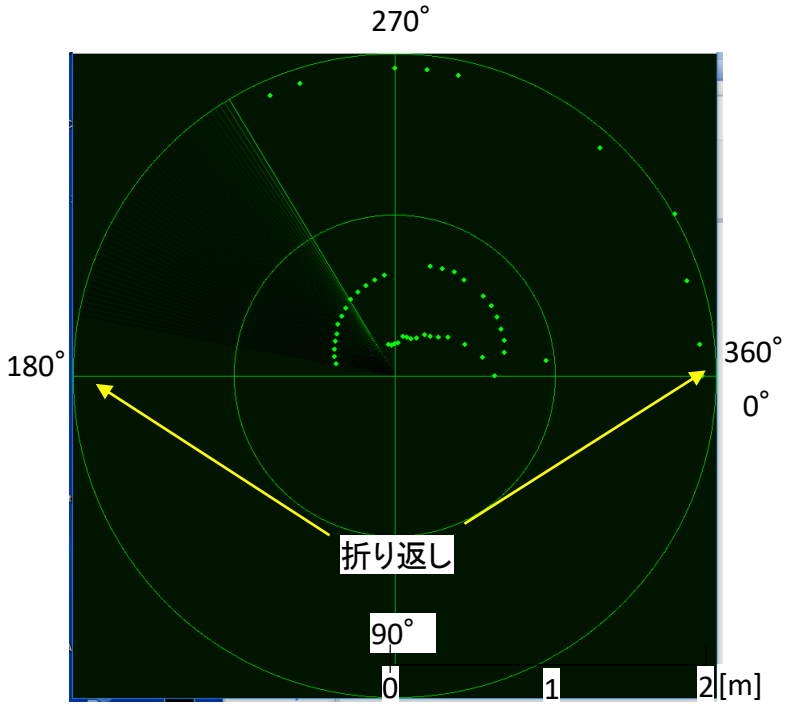
センサ回転方式を用いたPython (pygame)によるリアルタイムレーダ





レーダー表示

サーボモータによる回転

- 距離はバラつきが多いため，数回測定し，平均化
- 回転動作が安定後に測定実行とするために、サーボレート
  時間設定のためのdelay()関数は回転命令の直後に配置
- 180度回転時間, step 6度（30点）
  約2秒：40～50 cm、約4秒：200～400 cm
- 対象物への斜め照射は測定不可能
  応答時間の不揃いはこの方式の弱点

Arduino マイコン側

```
/* ******************************************
 * 2016.10.25(Tue.)  US-100  for RADAR
 * US-100 & AQM0802A == Distance & degrees
 * Non Timer
  USART ..Universal Shynchrounous Asyncrhrounous
        Receiver Transmitter
        汎用同期非同期式送受信
 ******************************************
 * 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

volatile const int LASER   = 8;
//const int trigger = 6;
volatile const int trigger = 4;
volatile const int echo    = 7;
volatile const int servono = 9;
volatile const int step    = 6;
volatile const float angle = 0;
volatile int cnt ;
volatile const float K = 1.15;    // at 23 [degrees]℃

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");
  delay(1000);
#endif
  Serial.begin(115200);
  //Serial.begin(9600);
  servo.attach(servono);

  //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]
  volatile float data;

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

void loop() {
  float deg  = 0;
  volatile const float stangle =  10;
  volatile const float offset  =  10;
  volatile const int servorate =  12;
  // const float enangle = 175;

  // 0-180 度回転
  for ( deg = stangle; deg < 180; deg ) {
    servo.write(180 - deg + offset);
    delay(servorate);
    Meas();
    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("cnt = ");
    lcd.print(cnt);
#endif

    digitalWrite(LASER, output);
    output = !output;
  }
  // 360-180 度回転
  for ( deg = 180; deg > stangle; deg ) {
    deg -= step;
    servo.write(180 - deg + offset);
    delay(servorate);
    Meas();
    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("cnt = ");
    lcd.print(cnt);

#endif
    digitalWrite(LASER, output);
    output = !output;
  }
}
```

Python (pygame) 側

データは有線（USB）経由で得るので文字化けや脱落などが極く少ないのでそのための処理は不要

```
# -*- coding: utf-8 -*-
#
# radar: RAdio Detecting And Ranging
# http://denshi.blog.jp/arduino/python_pygame_radar
#
# sweep 速度は送信側のrate、および通信速度に依存します.
# 超音波使用の場合、センサの処理速度，それらの制御
# マイコンの処理速度にも依りますが，音波の伝搬速度の方が
# 大きく，遠距離ほど遅い rate となると推定されます.
# 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、
# レーダーの近傍には近づかないなどの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("COM5",115200)  # COMポート(Arduino接続)

    i = 0
    while i <= 10 :    # 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)
        # (deg, L) = data.split(",")
        (L, deg) = data.split(",")     # mtaka

        # (deg, L) = (int(deg), int(L))
        (deg, L) = (int(deg), int(L)/5) # 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()
```