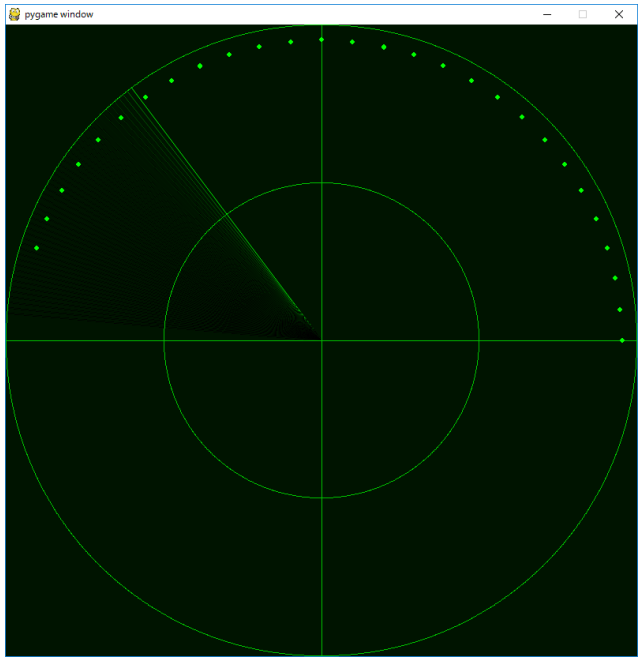
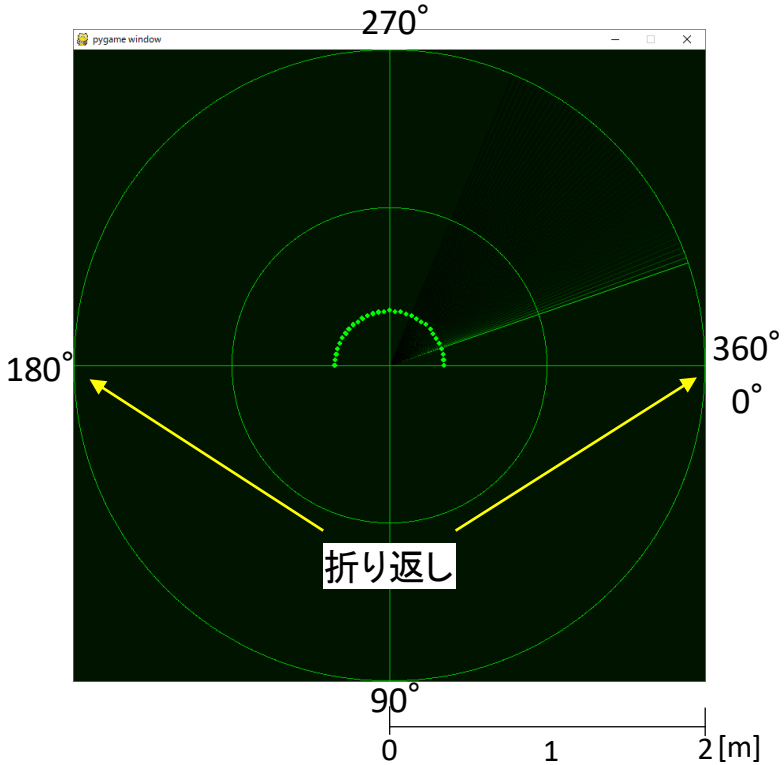
Python (pygame)によるUS-100を用いたレーダー表示

車のワイパーの様に動作させる（Arduino側の送信形式に依存）

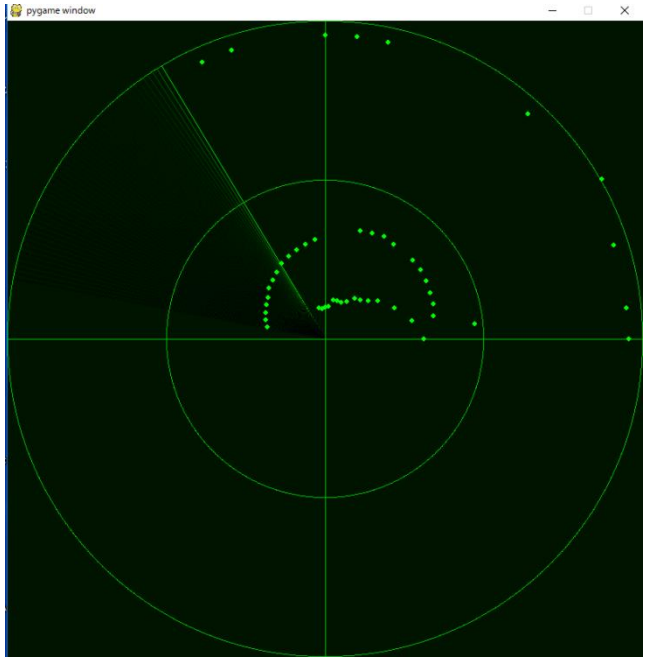
http://www.geocities.jp/mtakapii/Sonic.html



天井までの距離

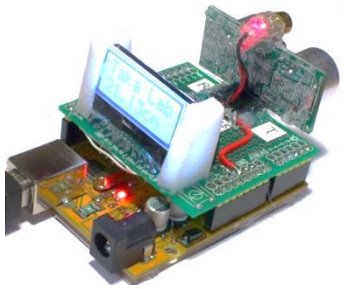


至近距離



対象物移動

- センサ回転角可変機構は持たず、回転角はマイコンにてソフト発生.
- 前方向にのみ進む移動物体への搭載を想定した信号形式.
- 距離が近いほど回転速度は速くなる（音速に依存したArduino側処理）.
- シリアルポート通信速度は，115200bps ，回転ステップは 6°
- 今後は，サーボモーターなどで回転させ，角度（方位）を同期させる予定.



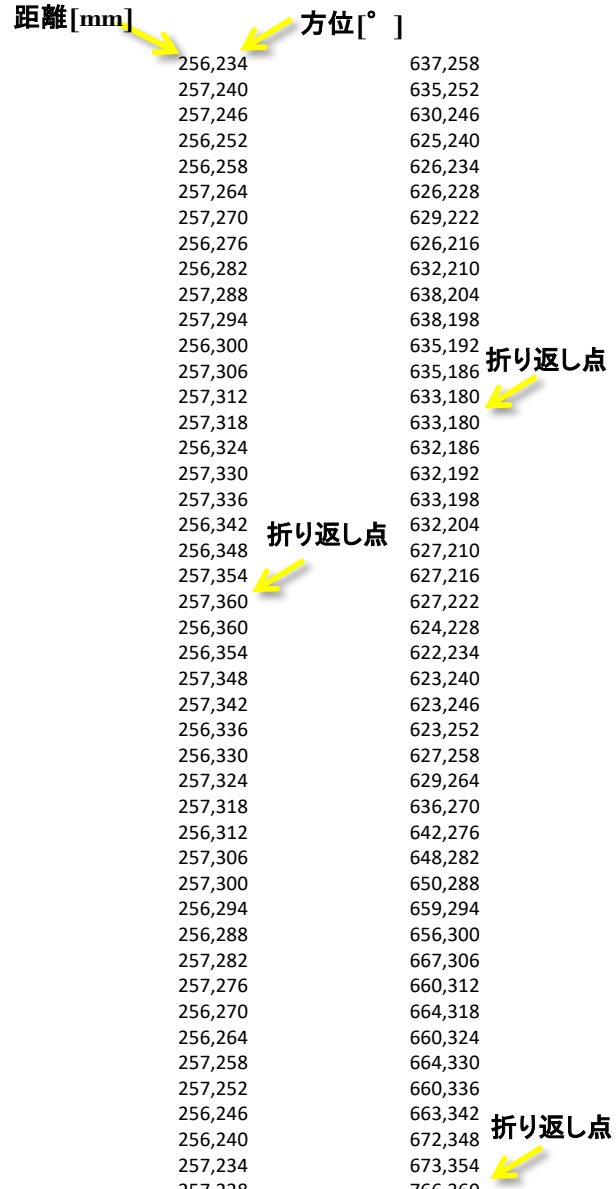
radar: RAdio Detecting And Ranging

Arduino マイコン側

```
/* ********************************************
 * 2016.10.14 mtaka  US-100
 * US-100 & AQM0802A == Distance & Velocity
 * Non Timer for radar Graph
 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

ST7032 lcd;
const int LASER   = 8;
//const int trigger = 6;
const int trigger = 4;
const int echo    = 7;

volatile int DST1;
volatile int DST2;
volatile float distance1;
volatile float distance2;
volatile float distance3;
volatile static boolean output = HIGH;

void setup() {
  lcd.begin(8, 2);
  lcd.setContrast(30);
  lcd.print("Taka Lab");
  lcd.setCursor(0, 1);
  lcd.print("US-100");

  Serial.begin(115200);
  // Serial.begin(9600);
  // Serial.begin(38400);
  Serial.println();
  // Serial.println("I'm 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 distance;

  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; //
    return (distance);
  }
  else{
    return(-1);
  }
}

void loop() {
  volatile int deg, stp, direct;
  volatile int Velo1, Velo2;

  direct = 1;
  stp = 6;
  deg = 180;
  while (true){
    distance1 = Meas();
    //delay(10);
    //distance2 = Meas();
    //delay(10);
    //distance3 = Meas();
    //Velo1 = (distance1 - distance2) * 100. / 36. ;
    //Velo2 = (distance2 - distance3) * 100. / 36. ;
    //Velo1 =   (Velo1 +   Velo2) / 2 ;
    DST1 =  distance1;
    Serial.print(DST1);
    //Serial.print(" mm  ");
    //Serial.print(",");
    Serial.print(",");
    lcd.setCursor(0, 0);
    lcd.print("       ");
    lcd.setCursor(0, 0);
    lcd.print(DST1);
    lcd.print(" mm");

    // Serial.println(Velo1);
    Serial.println(deg);
    deg += stp*direct;
    if ( deg > 360 ){
      deg = 360;
      direct = -1;
    }
    if ( deg < 180 ){
      deg = 180;
      direct = 1;
    }
    // Serial.println(" km");
    // lcd.setCursor(0, 1);
    // lcd.print("       ");
    // lcd.setCursor(0, 1);
    // lcd.print(Velo1);
    // lcd.print(0);
    // lcd.print(" km");

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

送信例

距離[mm]	方位[°]
256,234	637,258
257,240	635,252
257,246	630,246
256,252	625,240
256,258	626,234
257,264	626,228
257,270	629,222
256,276	626,216
256,282	632,210
257,288	638,204
257,294	638,198
256,300	635,192
257,306	635,186
257,312	633,180
257,318	633,180
256,324	632,186
257,330	632,192
257,336	633,198
256,342	632,204
256,348	627,210
257,354	627,216
257,360	627,222
256,360	624,228
256,354	622,234
257,348	623,240
257,342	623,246
256,336	623,252
256,330	627,258
257,324	629,264
257,318	636,270
256,312	642,276
257,306	648,282
257,300	650,288
256,294	659,294
256,288	656,300
257,282	667,306
257,276	660,312
256,270	664,318
256,264	660,324
257,258	664,330
257,252	660,336
256,246	663,342
256,240	672,348
257,234	673,354
257,228	766,360
256,222	680,360
257,216	676,354
257,210	681,348
256,204	677,342



python PC側

```
# -*- 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) = (800,800)  # 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
    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/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.display.update()       # 画面更新
        screen.fill((0, 20, 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()
```