XV-11 LIDAR

発表日 2016 年 4 月 28 日（木）

東京海洋大学 田原研究室 伊藤大智

発表内容

1. XV-11 LIDAR とは
2. 今回の課題の目標
3. 目標達成までのプロセス
   3-1.システム構成を考える
   3-2.回路設計・回路製作
   3-3. Octave でデータ処理
   3-4. Arduino LEONARDO でデータ処理
   3-5. Processing を用いてレーダ画面・GUI を作る
   3-6. アクリル板を置いたときの影響を調べる
   3-7. 障害物のない広い部屋での実験
4. 結果
5. 今後の展望
6. 参考文献

1. XV-11 LIDAR とは
   掃除ロボット「Neato XV-11」に使われている部品。赤外線レーザーが回転し周囲の障害物を検知。

図 1　Neato XV-11 (画像出典元 http://amzn.com/B003UBPB6E)

図 2 XV-11 LIDAR

2. 今回の課題の目標
目的の達成度を評価するための指標として、「サクセスレベル」を用いる。

ミニマムサクセス
XV-11 の出力データを Arduino を経由して PC で受信する。
XV-11 のモータを PWM 制御で回転させる。

フルサクセス
XV-11 の出力データを Arduino で処理し、決まったデータ出力フォーマットに変換して PC に送信する。
XV-11 を PC からコマンド操作できるようにする。

アドバンストサクセス
Processing を用いて PC 上にレーダ画面・GUI を作る
透明アクリル板を通しても距離データを取得できるか確認する

3. 目標達成までのプロセス

3-1. システム構成を考える

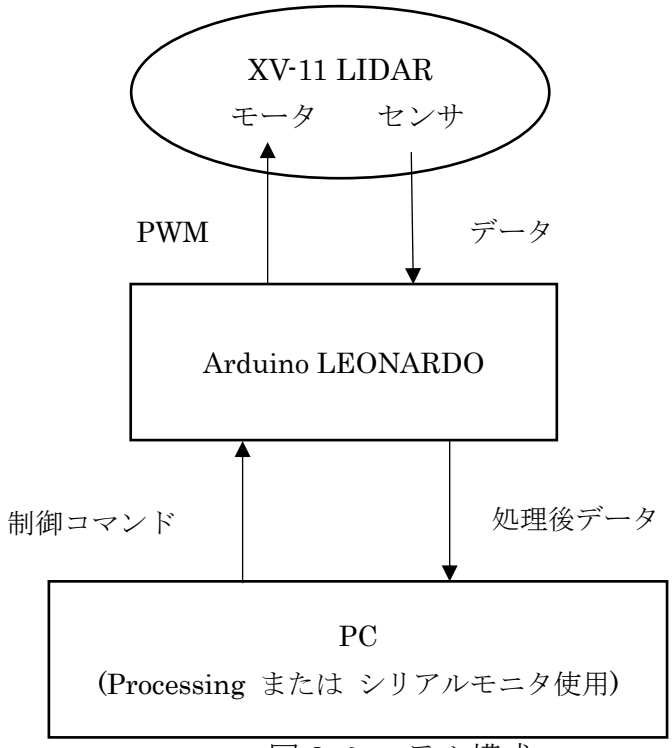

図 3 システム構成

・Arduino LEONARDO を使った理由
Arduino は XV-11 と PC 両方と同時にシリアル通信を行う必要がある。
XV-11 のボーレートは 115200bps。
Arduino は XV-11 とソフトウェアシリアルを使用してシリアル通信するわけだが、Arduino UNO や Arduino Duemilanove のソフトウェアシリアルは 115200bps に対応していない。
よって、Arduino LEONARDO を使用せざるを得なかった。

・モータの制御について
モータ制御は 12V, PWM 制御で行う

3-2.　回路設計・回路製作

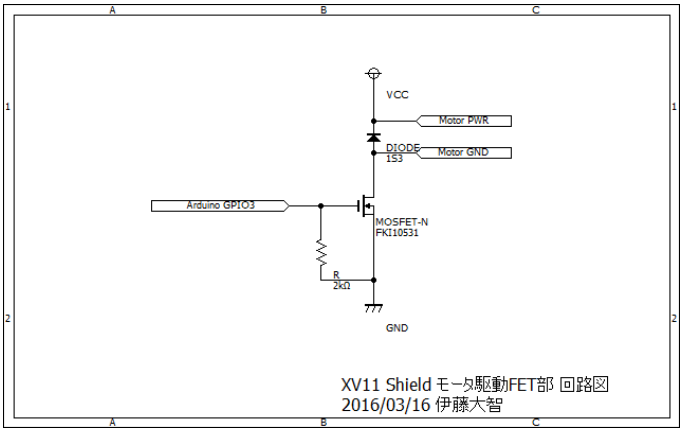

図 4　回路図

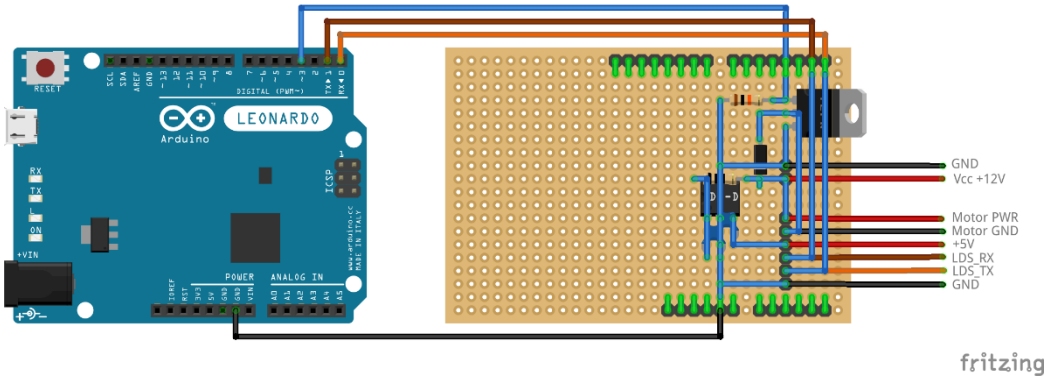

図 5　配線図（Top View）

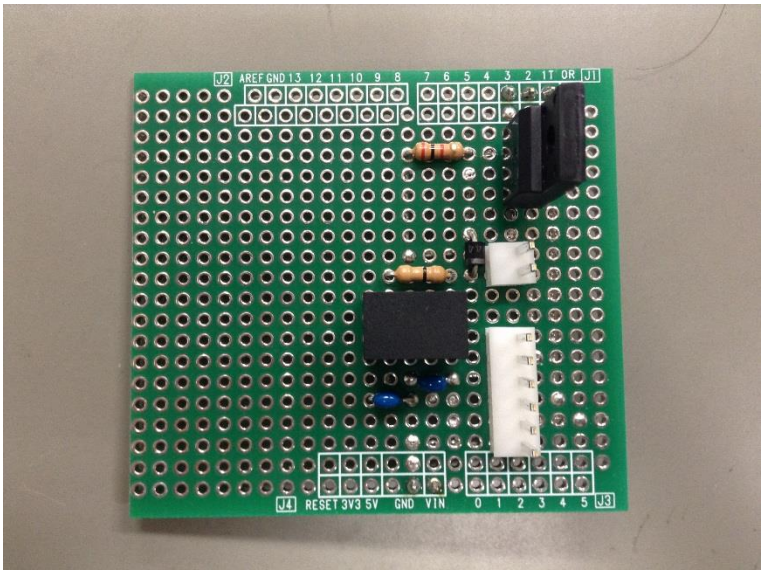

図 6　製作した回路

3-3. Octave でデータ処理

【XV-11 の出力データについて】
XV-11 は 1 回転につき、90 のパケットを送信する。
パケット 1 つあたりの長さは **22 バイト**。
パケット 1 つあたりに **4 個（角度 4 度分）のデータ**が含まれている。
　→ 4 度×90 パケット＝360 度

各パケットの構成は次の通り。

```
<スタート> <インデックス> <speed_L> <speed_H> [データ 0] [データ 1] [データ 2] [データ 3] <checksum_L> <checksum_H>
```

・スタートは、**常に 0xFA**
・インデックスは、0xA0 から 0xF9 の 90 通り（基準角度から何番目のパケットか表している）
・速度は 2 バイトで表される。<speed_H>の全 8 ビットと、<speed_L>の上位 2 ビットが整数を表し、<speed_L>の下位 6 ビットが小数を表す。

・ [データ 0] から[データ 3]は角度 4 度分の測定データ。データはそれぞれ 4 バイト長であり、4 バイトの構成は以下のようになっている。

バイト 0：距離データの下位 8 ビット
バイト 1：ビット 0 無効データフラグ
　　ビット 1 強さ警告フラグ
　　ビット 2-7 距離データの上位 6 ビット
バイト 2：信号強度の下位 8 ビット
バイト 3：信号強度の上位 8 ビット

距離情報はミリメートル単位であり、14 ビットで表される。
測定可能距離は 15cm から 6m である。
距離を計算することができなかった場合、無効データフラグか 1 になる。
測定距離に対して信号強度が期待されている値から大幅に劣っている場合、強さ警告フラグが 1 になる。

・チェックサムは 2 バイトで表される。受信したパケットのデータが正しいか判断するのに使う。XV-11 の場合、チェックに特殊なアルゴリズムを使う。

【Octave でデータ処理する前の準備】

・Arduino のプログラム
XV-11 からスタートビット(0xFA)を受信したら、スタートビットも含め 22 バイト（1 パケット）を受信し、PC に送信する

・XV-11 のモータ制御
Arduino から PWM 制御する。Duty 比を 75~95 あたりに設定して回転させると有効なデータが取れるということがわかった。

・実験方法
計測結果がわかりやすくなるよう、図 7 のように XV-11 をコンテナの中央に置いた。

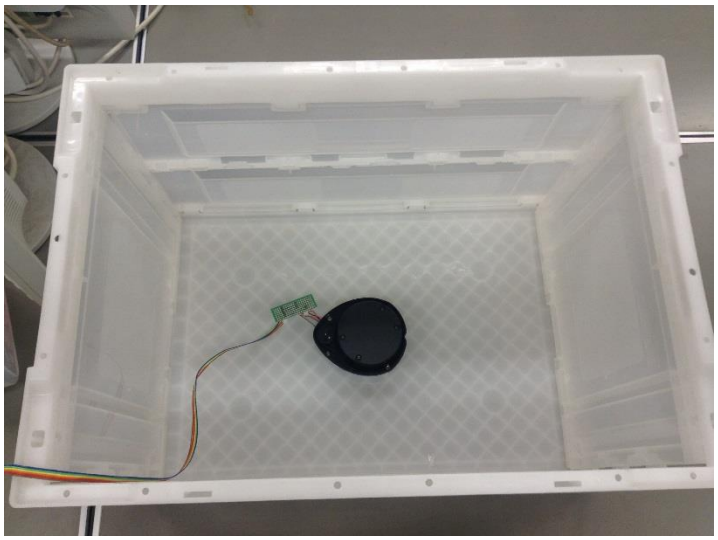

図 7　実験状況

パケットを 1 万回受信するまで XV-11 を回転させ、PC(TeraTerm 使用)で受信したデータは CSV ファイル形式で保存した。

この段階でミニマムサクセスはクリア

【Octave のプログラム作成】
CSV ファイルに保存された 22 バイト×10000 パケットをデータ処理するため、まず始めに以下の関数を作成した

function[ans_dec]=mystr2dec(data_str)
　data_str は 16 進表記 22 バイト分の文字列データ。
　これを数値データの配列(要素数 22)にして返す。

function[ans]=xv11_checksum(data_dec)
　data_dec の数値データ（要素数 22）からチェックサムをチェックする。
　チェックサムが正しければ 1, 違えば 0 を返す。

function[ans]=xv11_invalidflag(data_dec)
　data_dec の数値データ（要素数 22）の 6,10,14,18 番目の数値の先頭 1 ビットが invalid flag(無効フラグ)である。例えば 4 つすべて invalid flag が立っていたら[1,1,1,1]を返す。14,18 番目で invalid flag が立っていたら[0,0,1,1]を返す。

function[distance]=xv11_distance(data_dec,invalid_flag)
　data_dec の数値データ（要素数 22）から 4 つ（角度 4 度分）の距離データを取り出す。invalid flag が 1 のところは距離 0 とする。例えば、
　[0,197,0,198]というふうに返す。数値の単位は mm。

function[speed]=xv11_speed(data_dec)
　data_dec の数値データ（要素数 22）から回転速度データを取り出す。例えば、319.39 というふうに返す。単位は rpm。

function[angle]=xv11_angle(data_dec)
　data_dec の数値データ（要素数 22）の 2 番目の要素「インデックス」から角度を求める。例えば、[0,1,2,3]というふうに返す。単位は度（°）。

function=xv11_plot(result)
　配列 result は[i,angle,distance,speed]で構成される。（i は行番号。）angle と distance の情報から、xy 座標にプロットして図を作成する。
　配列 result について、checksum が正しくないときや invalid flag が立っているとき、angle,distance,speed は-1 となっている。

以上の関数を用いて、CSV ファイルをデータ処理し結果を図で表示する
　Octave プログラムを作成した。作成した Octave プログラムを以下に示す。
各関数ファイルは割愛する。

ファイル名：xv11_oct04.m

```
clear
pkg load all
more off


fp=fopen('C:/Users/Taichi/Dropbox/Octave/xv11/xv11_log/xv11_0013.csv','r')
i=1;

while((str=fgetl(fp)) != -1)
  time_str=str(1:30); # タイムスタンプは str の 1 文字目から 30 文字目
  [n,m]= size(str);
  dmy_str=str(32:m); # 32 文字目から最後までのデータを使う

  if rindex(dmy_str,',') !=0 # dmy_str の中に,が 1 つでもある場合
    data=strsplit(dmy_str,','); # 文字列 dmy_str を,で区切って小片に分割し，文字列配列と
  して data に代入

    count_num=str2double(data{1,1}) # いくつめのデータか
    data_str=data{1,2}; # xv11 の出力データ
    chksum_str=data{1,3}; # チェックサムは OK か NG か
    inv_num=str2double(data{1,4}); # 無効フラグの数
    inv_sum=str2double(data{1,5}); # 無効フラグの数の合計

    data_dec=mystr2dec(data_str); # xv11 の出力データを数値に変換

    if xv11_checksum(data_dec) # チェックサムが正しければ次の処理を行う
      speed=xv11_speed(data_dec);
      angle=xv11_angle(data_dec);
      invalid_flag=xv11_invalidflag(data_dec);
      distance=xv11_distance(data_dec,invalid_flag);
```

```
        for j=1:1:4
          if invalid_flag(j) # invalid flag が 1 であれば、result の各値は-1 とする
            result(i,:)=[i,-1,-1,-1];
            else
            result(i,:)=[i,angle(j),distance(j),speed];
          end
          i=i+1;
        end

        else #チェックサムが違うときは、result の各値は-1 とする
        for j=1:1:4
          result(i,:)=[i,-1,-1,-1];
          i=i+1;
        end
      end
    end
end

fclose(fp);

xv11_plot(result)
```

実行結果

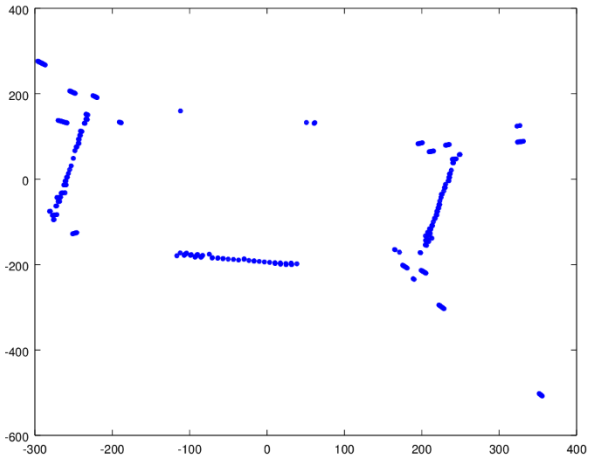

図 8　Octave でのデータ解析結果

3-4.　Arduino LEONARDO でデータ処理

・いままで Octave で行っていたデータ処理（座標プロット図の作成は除く）を、Arduino LEONARDO で行う。
・PC からのコマンド入力により、データ取得の開始・終了やモータの回転速度のコントロールをする。

【Arduino LEONARDO が行うデータ処理の流れ】

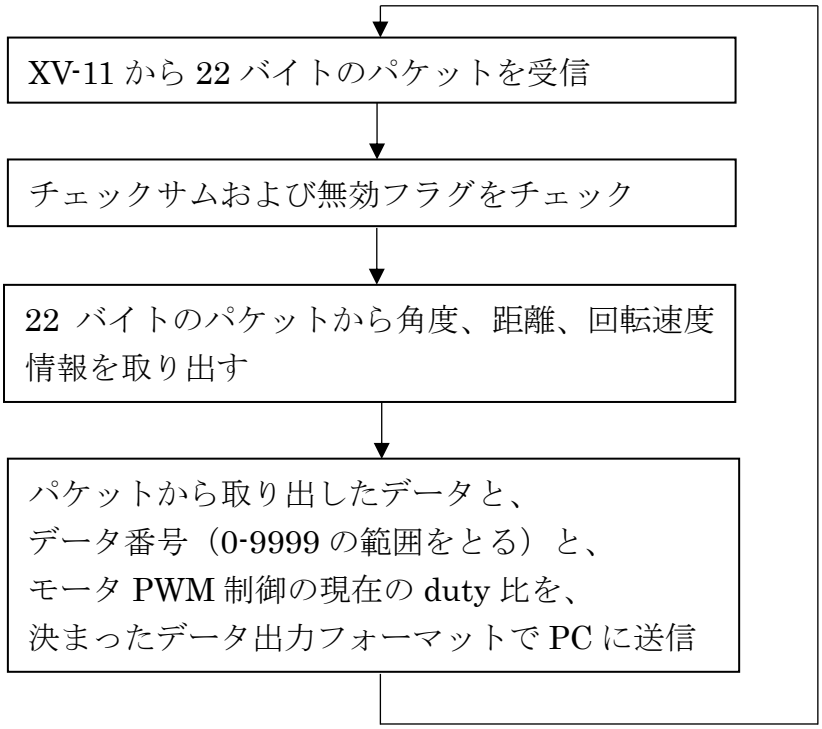

図 9　データ処理のフロー図

プログラムは長い(270 行)ので割愛

プログラムファイル名: xv11_13.ino
「最大 28,672 バイトのフラッシュメモリのうち、スケッチが 10,040 バイト（35%）を使っています。最大 2,560 バイトの RAM のうち、グローバル変数が 753 バイト（29%）を使っていて、ローカル変数で 1,807 バイト使うことができます。」Arduino IDE より

【データ出力フォーマット】
Arduino LEONARDO は、XV-11 が出力する 1 パケット 22 バイトのデータを処理し、角度情報(°)・距離情報(mm)・回転速度情報(rpm)を取り出し、ある決まったフォーマット（データ出力フォーマット）で出力するようプログラムされている。データ出力フォーマットは、次に示す出力データ例のように必ず 24 文字になる。

出力データ例
,9161,1,277,0158,322,075

各値の意味
9161:データ番号(0000-9999)
1: 距離データが有効であれば 1、無効であれば 0
277:角度(°) (000-359)
0158:距離(mm)
322:回転速度(rpm) (000-約 640)
075:PWM の duty 比(000-255)

データ出力フォーマットの先頭 1 文字および各値の間は , で区切られている。改行コードは'¥n'。先頭 1 文字に , がついている理由は、Teraterm で各行先頭につけられるタイムスタンプを分離するため。

【コマンド入力フォーマット】
Arduino LEONARDO は、シリアル通信で接続されているデバイスからある決まったフォーマット（コマンド入力フォーマット）のコマンドによって制御できるようプログラムされている。

コマンド入力フォーマット
␣,␣␣␣␣␣ 改行コード'¥r'(CR)
↑　　↑
記号　数値（4 桁まで有効・3 桁以下でも可）

記号には、s(スタート),e(エンド),p(duty 比設定),f(ファームウェア情報表示)の 4 種類がある。

表 10 各コマンドの説明

記号(小文字)	意味	内容
s	スタート	モータを回転させる。 入力された数値が 1~9999 の場合は、その数値+1 回データ取得する。 データ取得終了後、All end と表示してモータを停止する。 入力された数値が 0 の場合は、回数無制限でデータを取得し続ける。 数値が入力されなかった場合は、直前に設定された回数データを取得する。直前に設定された回数がない場合、回数無制限でデータを取得し続ける。
e	エンド	モータを停止する。数値は無効。
p	duty 比設定	入力された数値が 1~255 の場合は、その数値を PWM の duty 比に設定する。 それ以外の数値の場合は、duty 比は 0 に設定される。 duty 比が未設定または 0 の状態でスタートした場合、duty 比は初期設定値 75 に設定される。
f	情報表示	ソフトウェアのファイル名、作成年月日、作成者の名前を表示。
それ以外	無効	Your command was ignored と表示。

コマンド入力例

s,9999 ・・・10000 回データを取得する。
s,0 ・・・回数無制限でデータを取得し続ける。
s,99 ・・・100 回データを取得する。
s,0099 ・・・100 回データを取得する。
e ・・・モータを停止する。
p,80 ・・・duty 比を 80 に設定する。
p,080 ・・・duty 比を 80 に設定する。

この段階でフルサクセスまでクリア

3-5. Processing を用いてレーダ画面・GUI を作る

・Processing と、Processing の GUI を拡張するライブラリ controlP5 をインストールした。
・プログラムは田原 淳一郎 著『Arduino Uno/Leonardo で始める電子工作―8bit マイコンを活用するオープンプロジェクト Arduino の世界』に書かれているプログラムを参考にした。
・注意事項として、2016 年 3 月 22 日現在の最新バージョンの Processing と controlP5 では、本に書かれているプログラムは動作しなかった。本で使われているバージョンと同じものをインストールするとうまく動作した。

【作成した Processing スケッチの仕様】
・START ボタンを押すと s コマンドを送信
・END ボタンを押すと e コマンドを送信
・SPEED スライダで p,数値コマンドを送信（duty 比を変える）
・Arduino の出力データから読み取った Speed と PWM duty の値を画面に表示
・Arduino の出力データから読み取った angle と distance の値を元に座標を計算しプロット
・CLEAR ボタンを押すと画面のプロットをクリア
・ZOOM IN ボタンを押すとプロット距離の倍率が拡大
・ZOOM OUT ボタンを押すとプロット距離の倍率が縮小
・倍率を画面に表示
・プロットの原点座標に十字を表示
・画面の上方向がセンサの正面方向（基準角度方向）

作成した Processing スケッチの内容は長い(194 行)ので割愛
プログラムファイル名: xv11_p503.ino

図 11 は XV-11 をコンテナの中央に置き、start させてからしばらく待った後の画面

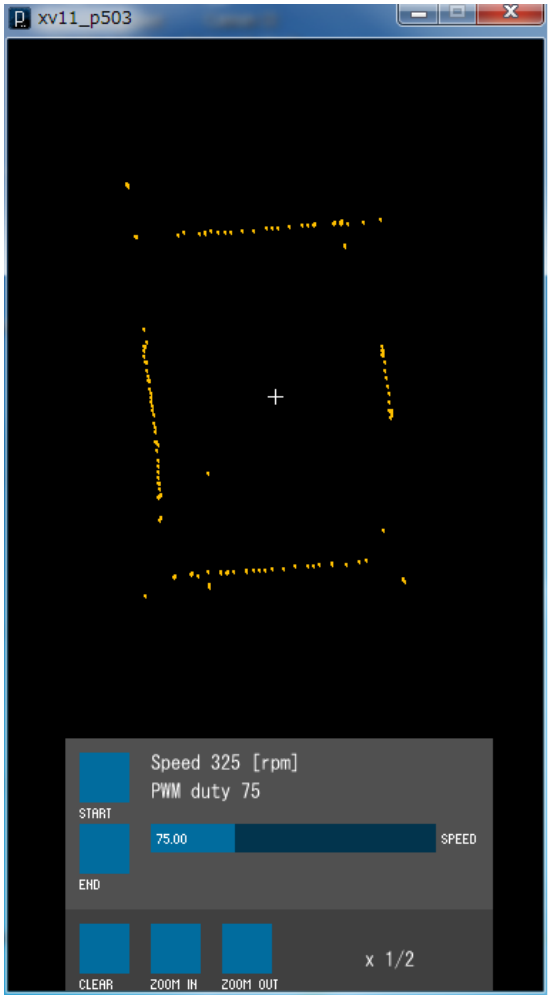

図 11　Processing 実行画面

3-6. アクリル板を置いたときの影響を調べる

水中機器に XV-11 を組み込む際、XV-11 をアクリル円筒の中に入れる必要がある。そこで、アクリル板を置いたときに距離データを取得できるかどうか実験で確認しておく必要があった。

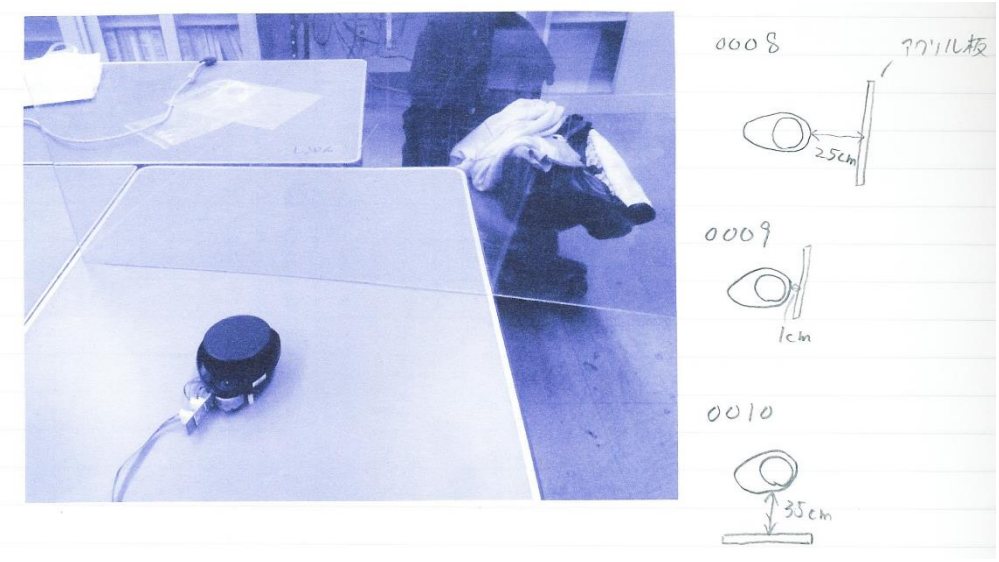

図 12　実験状況

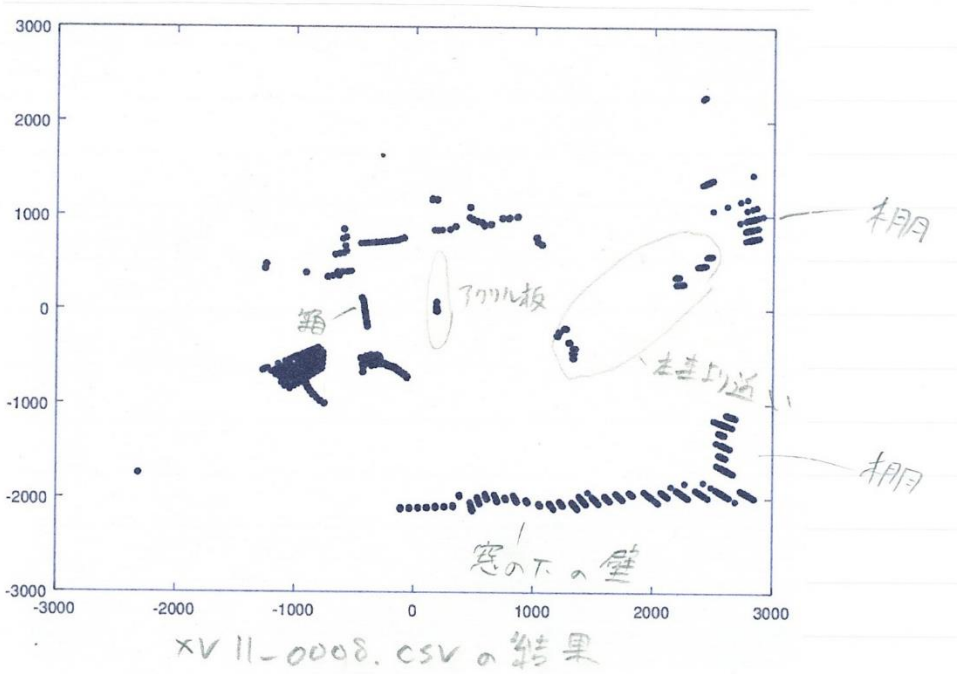

図 13　実験番号 0008 の結果

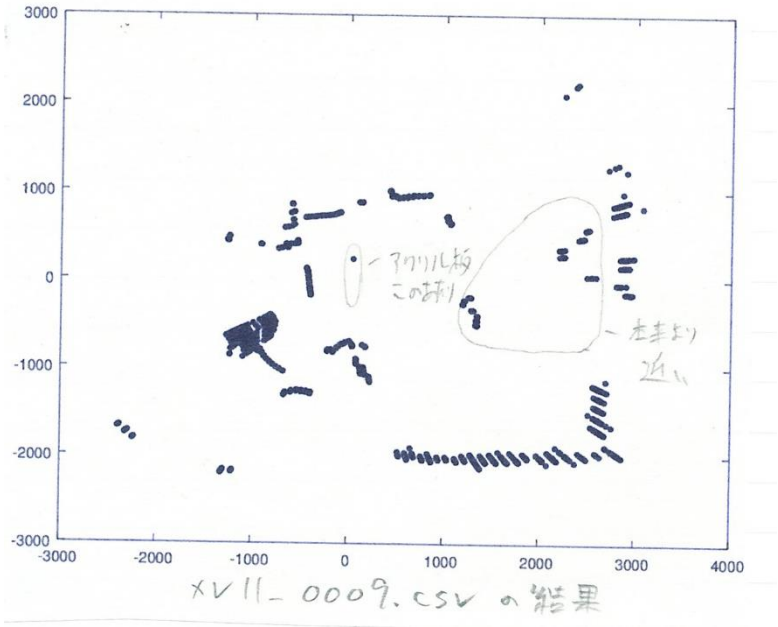

図 14　実験番号 0009 の結果

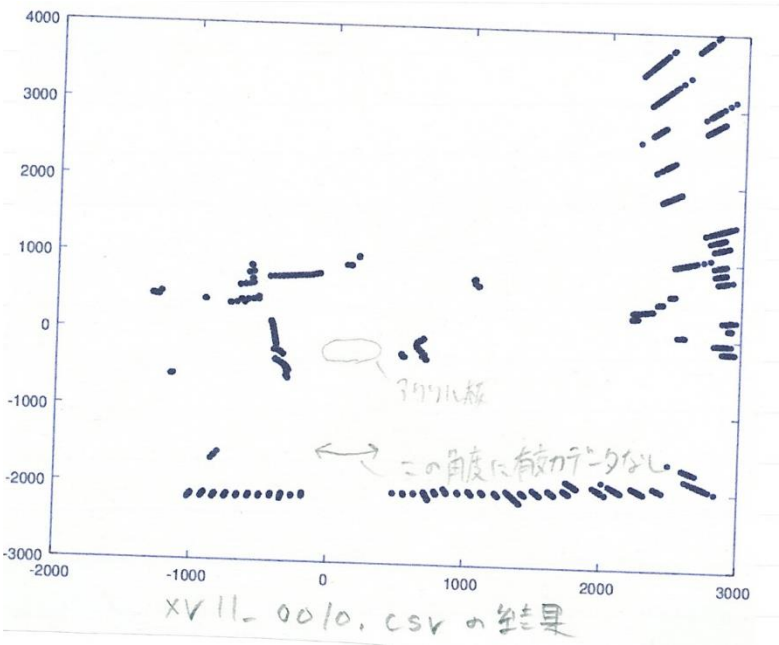

図 15　実験番号 0010 の結果

アクリル板があると、本来より近く距離を認識したり、有効なデータがとればくなったりすることがわかった。

この段階でアドバンストサクセスまでクリア

3-7. 障害物のない広い部屋での実験

XV-11 の測定可能距離は 15cm から 6m とされているが、本当に 6m まで測定できるのか確認する必要があった。そこで、四方を壁に囲まれ障害物のない部屋（暗室）で測定実験を行った。

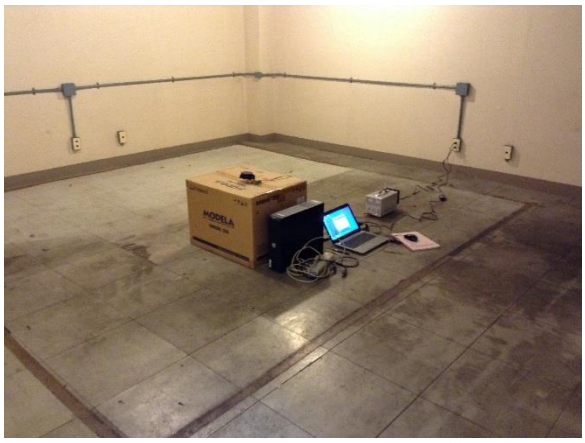

図 16　暗室での実験状況

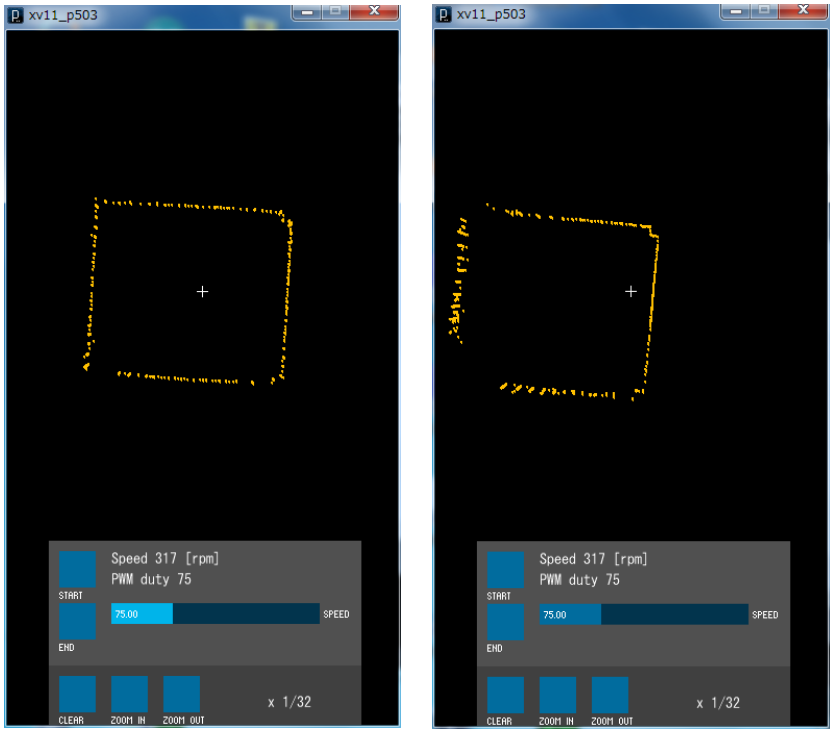

図 17 実験結果（左:暗室中央に設置した場合　右:暗室壁際に設置した場合）

暗室の広さは、6.0m×5.3m。XV-11 を壁際に設置したとき、6m より遠いところにある壁は測定できなかったことから、測定可能距離は最大 6m であることが確認できた。

4. 結果

・XV-11 制御回路(Arduino LEONARDO シールド)を設計・製作した。
・XV-11 のモータを PWM 制御で回転させた。
・XV-11 の出力データを Arduino を経由して PC で受信した。
・XV-11 の出力データを Octave でデータ処理した。
・XV-11 の出力データを Arduino で処理し、決まったデータ出力フォーマットに変換して PC に送信できるようにした。
・XV-11 を PC からコマンド操作できるようにした。
・Processing を用いてレーダ画面・GUI を作った。
・アクリル板を置いたときの影響を調べた結果、アクリル板があると本来より近く距離を認識したり、有効なデータがとれなくなったりすることがわかった。
・暗室での実験の結果、XV-11 が半径 6m の範囲まで計測可能であることを確認した。

5. 今後の展望

今後、卒業研究等で製作する水中機器に XV-11 を組み込む際、XV-11 をアクリル円筒の中に入れる必要がある。
XV-11 が入るアクリルパイプの選定を行い、アクリルパイプを発注し、XV-11 の防水ケースを製作する。

XV-11 のアクリルパイプの選定結果

【必要なアクリル円筒の条件】
内径φ140 以上
長さ　最低 50.5mm
スペーサ足をつけるなら 54mm
Arduino 基板を含めるなら 99mm

【結論】
はざいやでは、**12,040 円**
アクリヤドットコムでは、**13,437 円**
どちらも透明アクリルパイプ(押出し) 外径 200mm 厚さ 3mm　長さ 1000mm

6. 参考文献

xv11hacking – LIDAR Sensor
https://xv11hacking.wikispaces.com/LIDAR+Sensor
XV-11 LIDAR Sensor について基本的なことはすべてここに書かれていた。

XV_Lidar_Controller / XV_Lidar_Controller.ino – GitHub
https://github.com/getSurreal/XV_Lidar_Controller/blob/master/XV_Lidar_Controller.ino
Arduino コンパチブルボードで作った XV-11 LIDAR Sensor の Arduino プログラム。
似たようなことをやっているものの、あまり参考にはしなかった。

Read serial data from XV11 sensor – Arduino Stack Exchange
http://arduino.stackexchange.com/questions/18598/read-serial-data-from-xv11-sensor
XV-11 LIDAR Sensor から Arduino でデータ取得をしようとしてもうまくいかない人が質問をしているページ。

The Strange Storage: Arduino と Processing で遊ぶ！
http://www.storange.jp/2012/03/arduinoprocessing.html
Processing のスケッチについて、どうなっているのか意味が詳しく書かれていた。

Arduino Uno/Leonardo で始める電子工作―8bit マイコンを活用するオープンプロジェクト Arduino の世界
田原 淳一郎 (著)
Processing のプログラムを書く際に参考にした。