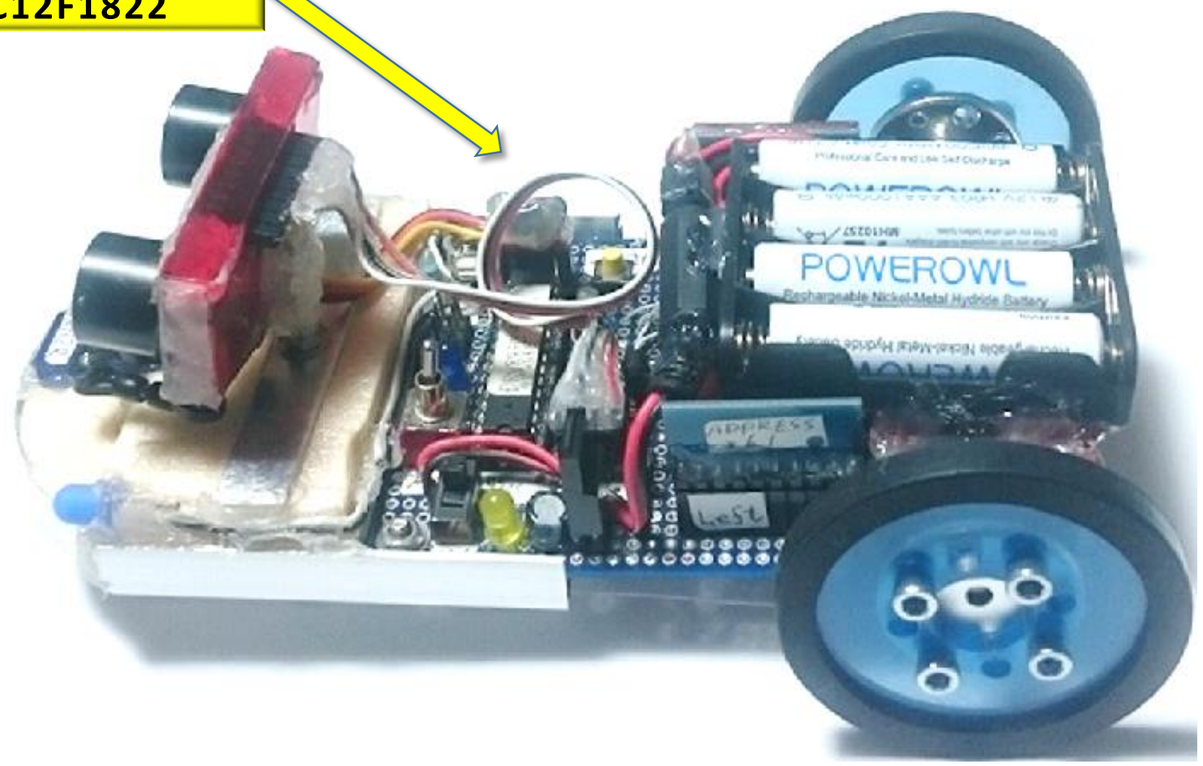
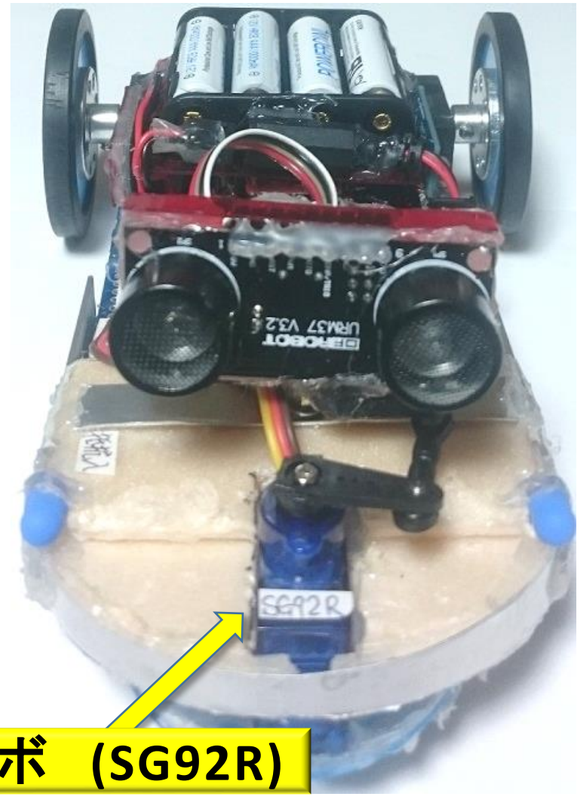
Atmega328P　DRV8830 x2　URM37 BMX055
障害物回避自動操縦の真似事

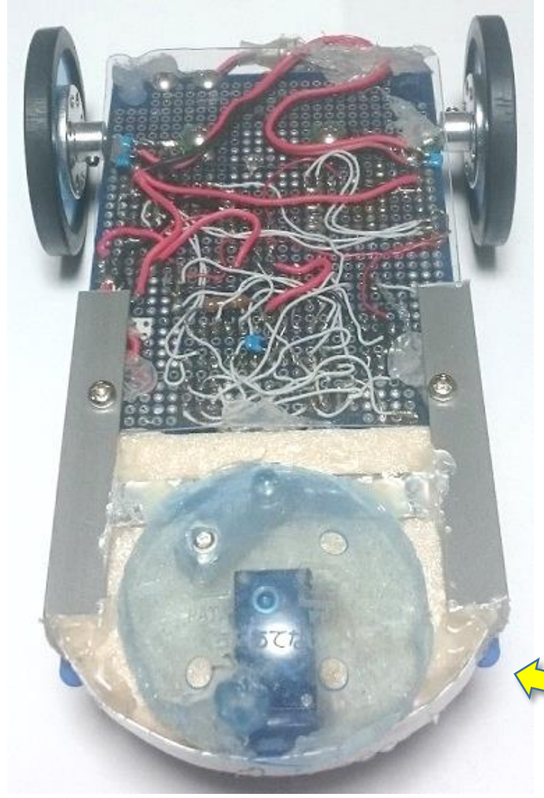
2021.1.29

距離センサー,
ステアリング
シリアルサーボ化
PIC12F1822



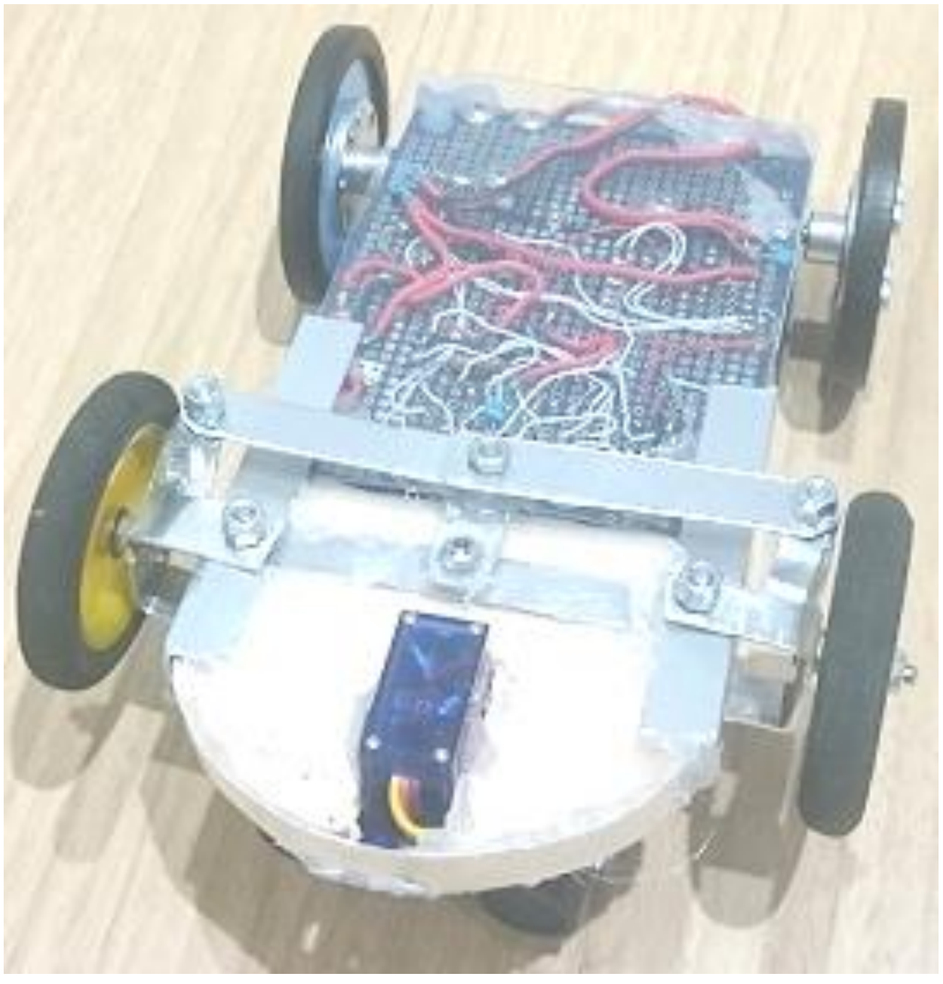


ラジコン用サーボ　(SG92R)



リセット時プログラムモード切替機能
アナログ入力端子A1の電圧値によって,
障害物回避か、一定距離追従型かの
プログラムモードを切り替える.
切り替えはタクトスイッチによって指定する.
超音波センサー円錐角は意外と広く，かなり近接の物にも反応する。なので、多少上向きに取り付けている.
電源は，単4タイプのNiMH 1000mAh　4本の電池で，通常使用には，電圧，容量ともに充分.

ガチャガチャ空ケース



ステアリングタイプに改造
材料:
0.5mm アルミ板　:　ステアリングロッド
1mm　アルミ板　:　主なステアリング部品
サーボモーター　:　超音波センサー首振り用と共用

2905L-3.3V
低損失
B A
\- +
3.53V
～7.4V
TA4805S
低損失
+5V
47uF
16V
100Ω
LED:RED
USB(B)
Top view
Now no use
+3.3V
入
入
出
+5V
DRV8830
G
G
#1
SCL
SDA
\-
+
+5V
10kΩ
10kΩ
+5V CPU
0.1uF
0.05Ω
0.1x2
2.4kΩ
1.3kΩ
ADXL345
+3.3V
BMX055
+5V CPU
GND
SDA
SCL
VCC
VCCIO
3V3
SDA
SCL
MOSI
MISO
2.4kΩ
1.3kΩ
#0
SCL
SDA
\-
+
+5V
0.1uF
0.05Ω
0.1x2
DRV8830
Vcc COMMON
RESET
ATMEGA328P
TX → RX D0
RX ← TX D1
INT 0 D2
INT 1 ~D3
D4
VCC
GND
XTAL
XTAL
~D5
~D6
D7
D8
RESET
A5 SCL
A4 SDA
A3
A2
A1
A0
GND
AREF
VCC
D13 SCK
D12 MISO
D11~ MOSI
D10~
D9~
2.3kΩ
低損失
4805F
5V500mA
+5V CPU
+5V CPU
22pF
16MHz
22pF
USBシリアル
変換モジュール
RX
GND
シリアルサーボ:差し替える
SG92R
PIX12F1822
黄
茶
赤
Sout:5
GND:8
2:RX
1:Vdd → 5V
+5V CPU
3.9kΩ
Tact
switch
cable
URM37 V3.2
黒 GND
白 PWM
緑 TRIG
赤 Vcc
CN
CN
LED:BLUE
300Ω
MISO
TEST
MOSI
WRITE
1 2
MISO
SCK
RST
VCC
MOSI
GND
ISP
TOP VIEW
MOSI
MISO
MOSI
MISO

通信ポート
DRV8830 : I2C
BMX055 : I2C
ADXL345 : SPI

初期段階では、USB(B)から電源供給したが、後の１バッテリーBA駆動を考慮し、
Atmega328P用に5Vと，ADXL345用に3.3Vレギュレータをそれぞれ実装した.
ただし、バッテリーとUSB(B)は同時接続してはいけない．LED用抵抗値は300Ω程度が適切.
ADXL345とmkII はSPI信号が競合するので，MISO,MOSI信号を切替え式とした.
ここでは，折角ADXL345が動作始めたこともあり，併用とした．更に距離センサーURM37も
追加.　　メモ：ADXL345は加速度センサー、BMX055は角加速度センサー

スケッチ ： DRV8830_2_Human_Robot_URM37_V4_steering.ino

```
// DRV8830_2_Human_Robot_URM37_V4_steering.ino
// 2021.1.29  mtakapii4@gmail.com
// ヒューマンアカデミーロボット向け自動操縦見たいな！
// for "Human Academy Robot Micon" Original Block
// use Serial servo (PIC12F1822)
/* ********************************************************/
// # Connection:
// #       Pin 1 VCC (URM V3.2)      -> VCC (Arduino)
// #       Pin 2 GND (URM V3.2)      -> GND (Arduino)
// #       Pin 4 PWM (URM V3.2)      -> Pin 3 (Arduino)
// #       Pin 6 COMP/TRIG (URM V3.2) -> Pin 5 (Arduino)
/* ********************************************************/
int URPWM = 3; // PWM Output 0－25000US, Every 50US represent 1cm
int URTRIG= 5; // PWM trigger pin
unsigned int Distance= 0;
uint8_t EnPwmCmd[4]={
 0x44,0x02,0xbb,0x01];   // distance measure command
/* ********************************************************/


#include <Wire.h>


// DRV8830 Motor Driver I2Cアドレス
#define DRV_ADR_0 0x64 // DRV8830のI2Cアドレス #0
//#define DRV_ADR_1 0x67 // DRV8830のI2Cアドレス #1
#define DRV_ADR_1 0x61 // DRV8830のI2Cアドレス #1


#define CTR_ADR  0x00 // CONTROLレジスタのサブアドレス
#define FLT_ADR  0x01 // FAULTレジスタのアドレス
#define CLEAR    0x80
// ブリッジ制御
#define M_STANBY B00  // スタンバイ
#define M_REVERSE B01  // 逆転
#define M_NORMAL B10  // 正転
#define M_BRAKE  B11  // ブレーキ


// 電圧定義
//#define MAX_VSET 0x15  // 1.69V
//#define MIN_VSET 0x09  // 0.72V
//#define MAX_VSET 0x50
//#define MAX_VSET 0x3F
#define MAX_VSET 0x30
#define MIN_VSET 0x03
#define HOLD_TIME 40    // [ms]
#define STEP_TIME 4     // [ms]
#define DISTANCE_LIMIT 15  // [cm]
#define DISTANCE_HOLD 10  // [cm]
byte DRV_ADR;


// 制御コマンド送信
int write_vset(byte vs, byte ctr, byte DRV_ADR){
 clear_fault(DRV_ADR);
 Wire.beginTransmission(DRV_ADR);
 Wire.write(CTR_ADR);
 Wire.write( ctr + (vs<<2) );
 return Wire.endTransmission();
}


void clear_fault(byte ADR) {
 Wire.beginTransmission(ADR);
 Wire.write(FLT_ADR);
 Wire.write(CLEAR); // Clear FALUT flag
 Wire.endTransmission();
}


int ang ;
int SELECT ;


#define LED_PIN_BLUE 9


#define ang_MAX   130
#define ang_L     ang_MAX
#define ang_MIN   39
#define ang_R     ang_MIN
#define ang_center 85
```

```
void setup() {
 Serial.begin(38400);
 pinMode( LED_PIN_BLUE, OUTPUT );
 Wire.begin();


 write_vset(00, M_NORMAL, DRV_ADR_0 );        // Stop
 write_vset(00, M_NORMAL, DRV_ADR_1 );


 delay(500);
 SELECT = analogRead(A1);
 Serial.print("a");
 Serial.println(ang_center);
 delay(500);

 Serial.print("a");
 Serial.println(ang_MIN);
 delay(500);
 Serial.print("a");
 Serial.println(ang_MAX);
 delay(500);

 for( ang = ang_MIN ; ang < ang_MAX ; ){
   Serial.print("a");
   Serial.println(ang);
   ang += 1;
   delay(40);
 }
 for( ang = ang_MAX ; ang > ang_MIN ; ){
   Serial.print("a");
   Serial.println(ang);
   ang -= 1;
   delay(40);
 }
 Serial.print("a");
 Serial.println(ang_center);
 delay(500);

 PWM_Mode_Setup(); // URM37 V3.2
}

void PWM_Mode_Setup()
{
 pinMode(URTRIG,OUTPUT);            // A low pull on pin COMP/TRIG
 digitalWrite(URTRIG,HIGH);         // Set to HIGH
 pinMode(URPWM, INPUT);             // Sending Enable PWM mode command
 for(int i=0;i<4;i++)
 {
   Serial.write(EnPwmCmd[i]);       // URM37 V3.2
 }
}
void PWM_Mode()
{                  // a low pull on pin COMP/TRIG triggering a sensor reading
 digitalWrite(URTRIG, LOW);
 digitalWrite(URTRIG, HIGH);              // reading Pin PWM will output pulses
 unsigned long DistanceMeasured=pulseIn(URPWM,LOW);
 if(DistanceMeasured>=10200)
 {            // the reading is invalid.
   Distance = 150;
 }
 else
 {
   Distance=DistanceMeasured/50;     // every 50us low level stands for 1cm
 }
}

void loop() { //
 write_vset(00, M_NORMAL, DRV_ADR_0 );
 write_vset(00, M_NORMAL, DRV_ADR_1 );
 delay(300);
 // 順方向 徐々にスピードを上げて、下げる
 for (byte i = MIN_VSET; i <= MAX_VSET; i++) {
   write_vset(i, M_NORMAL, DRV_ADR_0 );
   write_vset(i, M_NORMAL, DRV_ADR_1 );
   delay(STEP_TIME);
 }


 delay(HOLD_TIME);
 for (byte i = MAX_VSET; i >= MIN_VSET; i--) {
   write_vset(i, M_NORMAL, DRV_ADR_0 );
   write_vset(i, M_NORMAL, DRV_ADR_1 );
   delay(STEP_TIME);
 }
 //digitalWrite( LED_PIN_BLUE, LOW );
 // 逆方向 徐々にスピードを上げて、下げる
 for (byte i = MIN_VSET; i <= MAX_VSET; i++) {
   write_vset(i, M_REVERSE, DRV_ADR_0 );
   write_vset(i, M_REVERSE, DRV_ADR_1 );
   delay(STEP_TIME);
 }
```

```
 delay(HOLD_TIME);
 for (byte i = MAX_VSET; i >= MIN_VSET; i--) {
   write_vset(i, M_REVERSE, DRV_ADR_0 );
   write_vset(i, M_REVERSE, DRV_ADR_1 );
   delay(STEP_TIME);
 }


 // MAX_VSET 左右連続旋回
 Serial.print("a");
 Serial.println(ang_L);
 write_vset(MAX_VSET/5, M_NORMAL, DRV_ADR_0 );
 write_vset(MAX_VSET, M_NORMAL, DRV_ADR_1 );
 delay(500);
 write_vset(MAX_VSET/5, M_REVERSE, DRV_ADR_0 );
 write_vset(MAX_VSET, M_REVERSE, DRV_ADR_1 );
 delay(500);


 Serial.print("a");
 Serial.println(ang_R);
 write_vset(MAX_VSET, M_NORMAL, DRV_ADR_0 );
 write_vset(MAX_VSET/5, M_NORMAL, DRV_ADR_1 );
 delay(500);
 write_vset(MAX_VSET, M_REVERSE, DRV_ADR_0 );
 write_vset(MAX_VSET/5, M_REVERSE, DRV_ADR_1 );
 delay(500);


 write_vset(00, M_NORMAL, DRV_ADR_0 );
 write_vset(00, M_NORMAL, DRV_ADR_1 );


 Serial.print("a");
 Serial.println(ang_center);


 int sensorValue;
 int Right_D;
 int Left_D ;


 if ( SELECT > 500 ){
   while(1){
     digitalWrite( LED_PIN_BLUE, HIGH );
     write_vset(MAX_VSET/2, M_NORMAL, DRV_ADR_0 );
     write_vset(MAX_VSET/2, M_NORMAL, DRV_ADR_1 );


     PWM_Mode();
     if( Distance < DISTANCE_LIMIT ){
       digitalWrite( LED_PIN_BLUE, LOW );
       write_vset(00, M_BRAKE, DRV_ADR_0 );
       write_vset(00, M_BRAKE, DRV_ADR_1 );
       delay(100);
       write_vset(MAX_VSET/2, M_REVERSE, DRV_ADR_0 ); // Back
       write_vset(MAX_VSET/2, M_REVERSE, DRV_ADR_1 );
       delay(600);
       write_vset(00, M_BRAKE, DRV_ADR_0 );         // Stop
       write_vset(00, M_BRAKE, DRV_ADR_1 );
       delay(200);


       Serial.print("a");
       Serial.println(ang_R);
       delay (400);
       PWM_Mode();
       Right_D = Distance;
       delay(100);


       Serial.print("a");
       Serial.println(ang_L);   // set MAX angle
       delay (400);
       PWM_Mode();
       Left_D = Distance;
       delay(100);


       if( Right_D < Left_D ){       // Turn Left
         Serial.print("a");
         Serial.println(ang_R);
         delay (200);
         write_vset(MAX_VSET, M_REVERSE, DRV_ADR_0 ); // Back
         write_vset(MAX_VSET/2, M_REVERSE, DRV_ADR_1 );
         delay(400);
         Serial.print("a");
         Serial.println(ang_center);
         delay (200);
         write_vset(00, M_NORMAL, DRV_ADR_0 );         // Stop
         write_vset(00, M_NORMAL, DRV_ADR_1 );
         delay(100);
       }
       else{
         Serial.print("a");
         Serial.println(ang_L);
         delay (200);
         write_vset(MAX_VSET/2, M_REVERSE, DRV_ADR_0 ); // Back
         write_vset(MAX_VSET, M_REVERSE, DRV_ADR_1 );
         delay(400);
         Serial.print("a");
         Serial.println(ang_center);
         delay (200);
         write_vset(00, M_NORMAL, DRV_ADR_0 );         // Stop
         write_vset(00, M_NORMAL, DRV_ADR_1 );
         delay(100);
       }
     }
     Serial.print("a");
     Serial.println(ang_center);
     delay(200);
   }
 }
 else{
   while(1){
     int VSET;


     PWM_Mode();
     if( (DISTANCE_HOLD-1) < Distance && Distance < (DISTANCE_HOLD +1)){
       write_vset(00, M_NORMAL, DRV_ADR_0 );
       write_vset(00, M_NORMAL, DRV_ADR_1 );
     }
     if( Distance < (DISTANCE_HOLD-1)){
       VSET = (DISTANCE_HOLD - Distance)*20;
       if(VSET > MAX_VSET){
         VSET = MAX_VSET;
       }
       write_vset(VSET, M_REVERSE, DRV_ADR_0 );
       write_vset(VSET, M_REVERSE, DRV_ADR_1 );
     }
     if( DISTANCE_HOLD < Distance){
       VSET = ( Distance - DISTANCE_HOLD)*20 ;
       if(VSET > MAX_VSET){
         VSET = MAX_VSET;
       }
       write_vset(VSET, M_NORMAL, DRV_ADR_0 );
       write_vset(VSET, M_NORMAL, DRV_ADR_1 );
     }
     delay(20);       // delay in between reads for stability
   }
 }
}
```