

令和三年度

応用制御工学特論

自律走行力一製作 最終報告書

提出日：2022 年〇月〇日

提出期限：2022 年 2 月 10 日

521M213

枘田優希

I. コンセプト

私の自律走行カーはコンセプトとして、「我々の研究室で用いられているセンサを活用し、コンパクトかつ安定した走行を行う自律走行カー」を設定した。具体的には、長距離の計測を可能とする LiDAR をサーボモータによって回転させることで、1つのセンサのみで安定した走行を可能とする車両の開発を試みた（図1）。

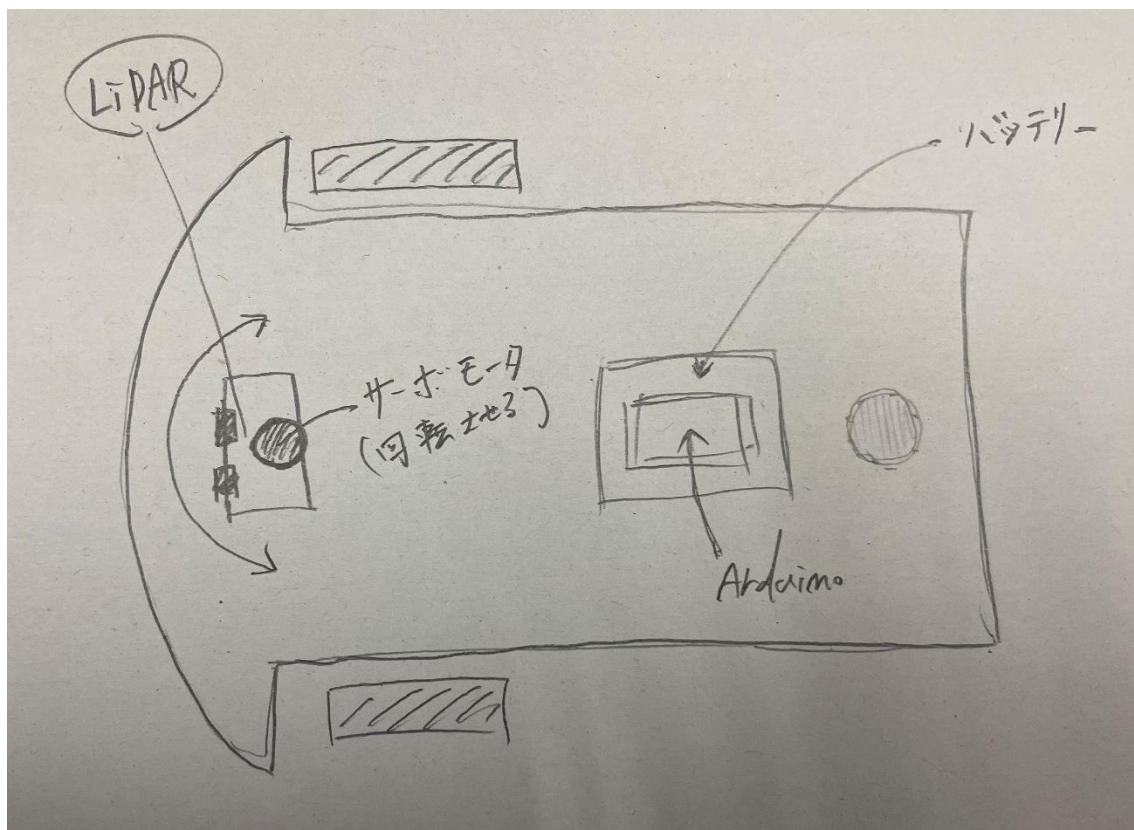












図1. 自律走行カーの概略図











Ⅱ. マイルストーン


(a) 計画初期のマイルストーン


工程	詳細	10 月 上旬	10 月 下旬	11 月 上旬	11 月 下旬	12 月 上旬	12 月 下旬	1 月 上 旬	1 月 下 旬
車 体 設計※ 1	概略図								
	図面設計 (CAD)								
	制作 (3D プ リ ン タ)								
	回路設計								
ア ル ゴ リ ズ ム 設計	経路計画 (センサ 情報の取 得+走行)								
車 体 組立	センサ, マイコン 等の固定								
プ ロ グ ラ ム 制 作	センサの 動作確認								
	コード記 述								
プ ロ グ ラ ム 修 正									
テ ス ト									

※1 車体設計（図面設計）→LiDAR とサーボ間の固定部品の設計

(b) 最終マイルストーン

工程	詳細	10 月上 旬	10 月下 旬	11 月上 旬	11 月下 旬	12 月上 旬	12 月下 旬	1 月上 旬	1 月下 旬
車体設計	概略図								
	図面設計 (CAD)								
	制作 (3D プリンタ)								
	回路設計								
アルゴ リズム 設計	経路計画 (センサ情 報の取得+ 走行)								
車体組 立	センサ, マイコン 等の固定								
プログ ラム制 作	センサの 動作確認								
	コード記 述+試走								
プログ ラム修 正									
テスト									

※  …初期の計画

※  …進捗状況

Ⅲ. 部品について

材料について, 表 1 に使用した部品とその価格を示す. マイコンには, ELEGOO の Arduino 用 Uno (図 2 (a)) を用いた. これは, 正規の Arduino に比べて安価で購入することが可能である. マイコンへの電源供給には, 5V/2A のモバイルバッテリー (図 2 (b)) を用いた. 距離を計測するセンサには, RCmall の TF-Luna LiDAR (図 2 (c)) を用いた. このセンサは, 最大 8m 先の物体までの距離を計測することが可能である. LiDAR を回転させる際に用いるサーボモータには, TOWER PRO の SG90 (図 2 (d)) を用いた.

表 1. 部品部とその価格

サイト名	名称	メーカー	型番	単価	数量	金額※	実金額		備考
amazon	ELEGOO Arduino用UNO R3コントロールボード ATmega328P +USBケーブル	ELEGOO	JP-EL-CB-001	1349	1	1456.92	1338	法人価格	マイコン部
amazon	モバイルバッテリー	Lakko		2320	1	2505.6	2420		電源供給用
amazon	TF-Luna Lidar距離計センサー	RCmall	TF-Luna	3099	1	3346.92	3099		計測用
amazon	デジタル・マイクロサーボ	TOWER PRO	SG90	440	1	475.2	540	送料+100	LiDAR回転用
	PLA樹脂			4	4.3	17.2			LiDAR固定部品用
	モータドライバ	TOSHIBA	TA7291P		2				
						合計 (税込)	7414.2		



(a) マイコンボード



(b) モバイルバッテリー

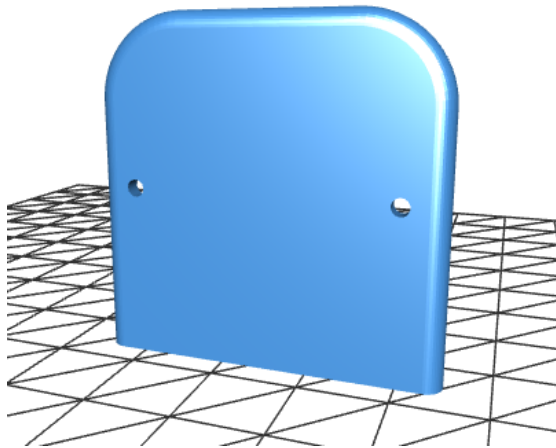


(c) LiDAR

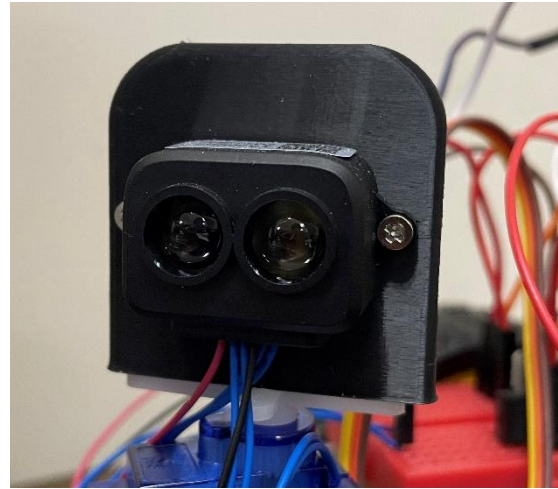


(d) サーボモータ

図 2 自律走行カーに搭載した部品



(b). 固定部の概観図



(c). LiDAR を固定した様子

図 3. LiDAR を固定する部品

(b) 電源

本自律走行カーでは、マイコン電源に 5V/2A のモバイルバッテリーを使用し、モータ電源に単三電池 4 つ (6V) を使用した。

(c) 電子回路

図 4 に本自律走行カーの全体回路図を示す。今回、サーボモータを動かす際にライブラリ (Servo.h) を使用した。このライブラリをインクルードすると、Arduino 用 Uno の 9 番ピンと 10 番ピンを同時に用いることが出来ないため、これらのピン以外 (今回の場合、5 番ピンと 6 番ピン) に DC モータの PREF (可変電圧入力) ピンを挿入した。

また、今回プログラムを作成するにあたり、Arduino IDE と Visual Studio Code の 2 種類のソフトウェアを用いた。

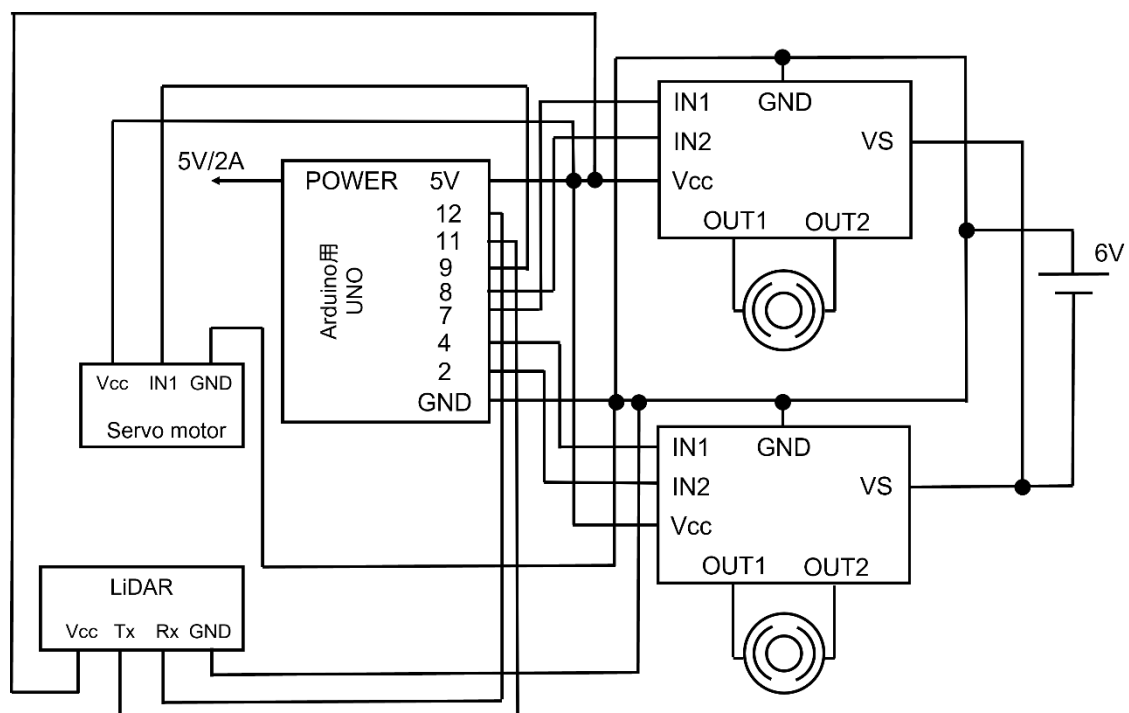


图 4. 全体回路图

V. 自律走行カーの制御

図 5 に走行ルートと各箇所における処理を表した図を示す。壁間の距離は 2m であることから、壁との距離に応じて左右タイヤの回転速度（PWM 値）を変化させる処理を行う（図 6、図 7）。左の壁沿いを走行する際の左のタイヤの速度を式 (1) に、右のタイヤの速度を式 (2) に示す。また、右の壁沿いを走行する際の左のタイヤの速度を式 (3) に、右のタイヤの速度を式 (4) に示す。 x は壁との距離、 y は PWM 値、 A は PWM 値の最大値、 B は PWM 値の最小値を示す。

まず、車体は、図 8 の①地点からスタートし、その際にサーボモータと LiDAR は図 6 に示すように、壁に対して斜め（ 147° ）に傾けて走行する。そして、壁との距離が 2.5 m 以上になる②地点を曲がり角であると定義し、車体を左に傾けながら走行する処理を行う。そして、ある一定時間、壁との距離が 2.5 m より近い値を取得することが出来れば、③地点のようにサーボモータを右（ 40° ）に傾けて、右の壁との距離を計測しながら走行を行う。次に、④地点に到達すると、壁との距離が 2.5 m 以上となるため、サーボモータを再び左（ 147° ）に傾けて、車体を左に傾けながら走行を行う。そして、⑤地点のように左の壁との距離を計測しながら走行を行う。3 つ目の曲がり角である⑥地点では、サーボモータの向きを 147° に固定した状態で車体を左に傾けながら走行を行う。⑦、⑧、⑨地点においては、⑤、⑥地点の処理と同様であるため割愛する。

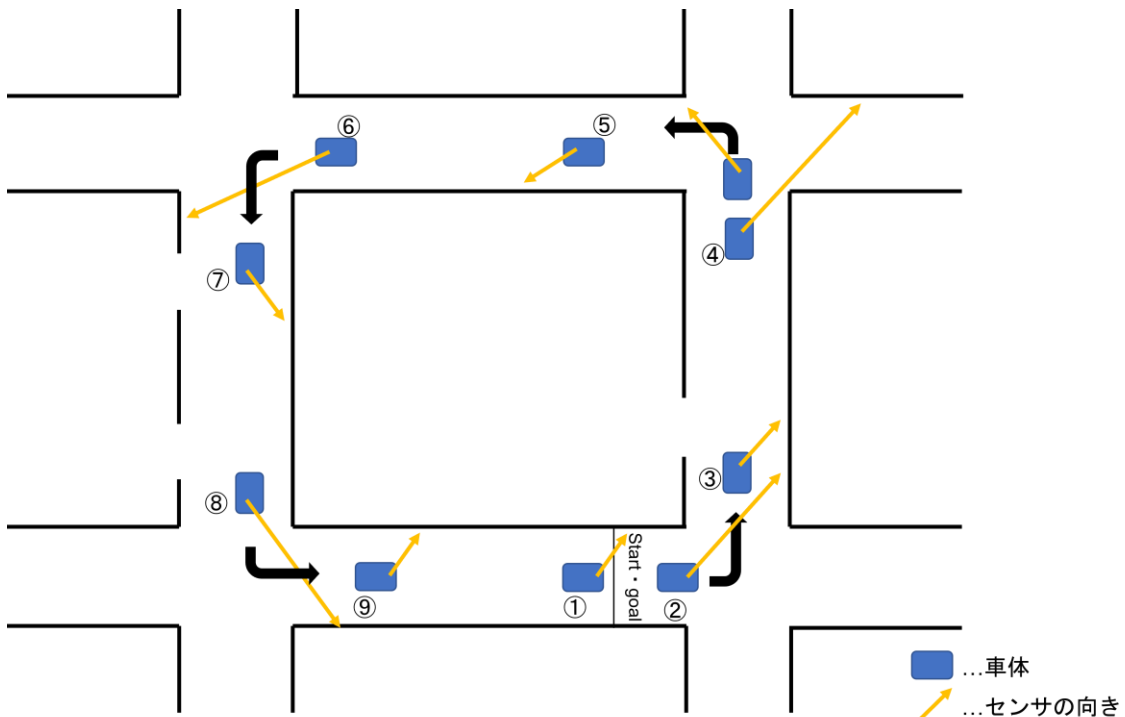
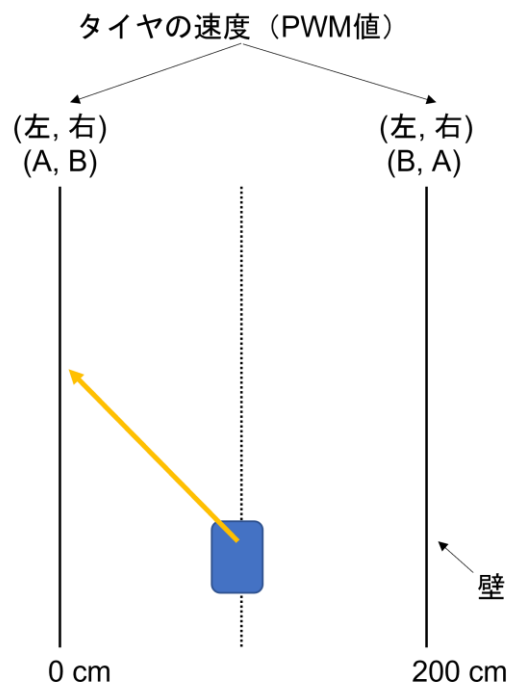
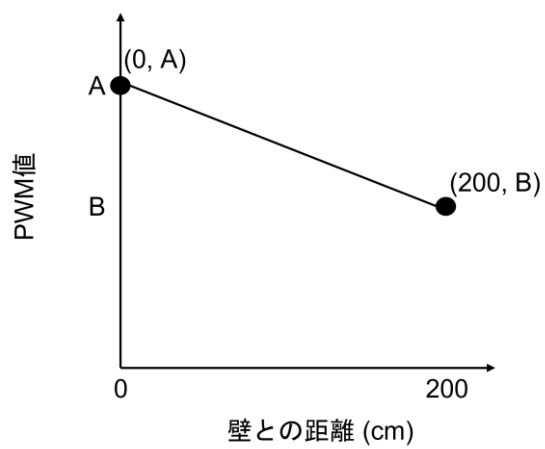


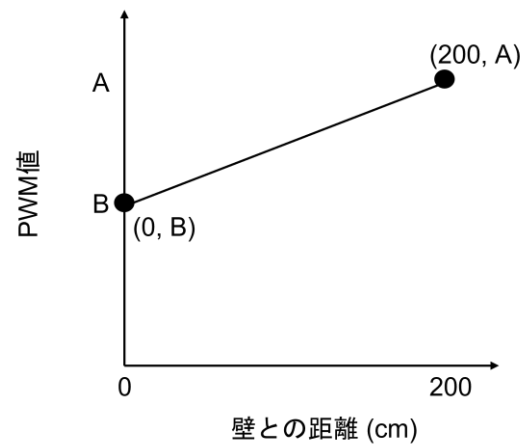
図 5. 自律走行カーの走行ルートと各箇所での処理を表した図



(a) 壁との距離と速度を表した図



(b) 左のタイヤの速度

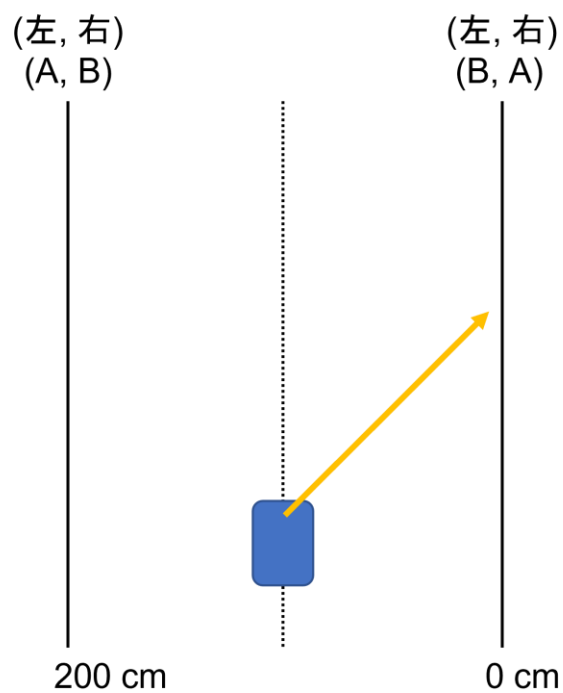


(c) 右のタイヤの速度

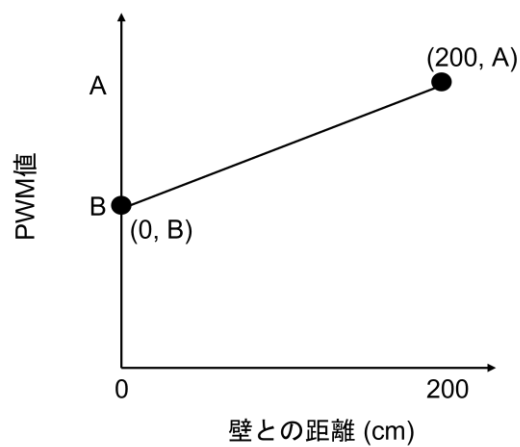
図 6. 左の壁沿いを走行する際の速度変化を表した図

$$y = \left(\frac{B-A}{200} \right) x + A \quad (1)$$

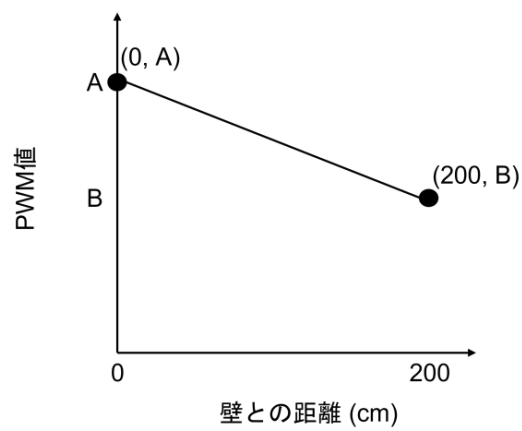
$$y = \left(\frac{A-B}{200} \right) x + B \quad (2)$$



(b) 壁との距離と速度を表した図



(b) 左のタイヤの速度



(c) 右のタイヤの速度

図 7. 右の壁沿いを走行する際の速度変化を表した図

$$y = \left(\frac{A-B}{200} \right) x + B \quad (3)$$

$$y = \left(\frac{B-A}{200} \right) x + A \quad (4)$$

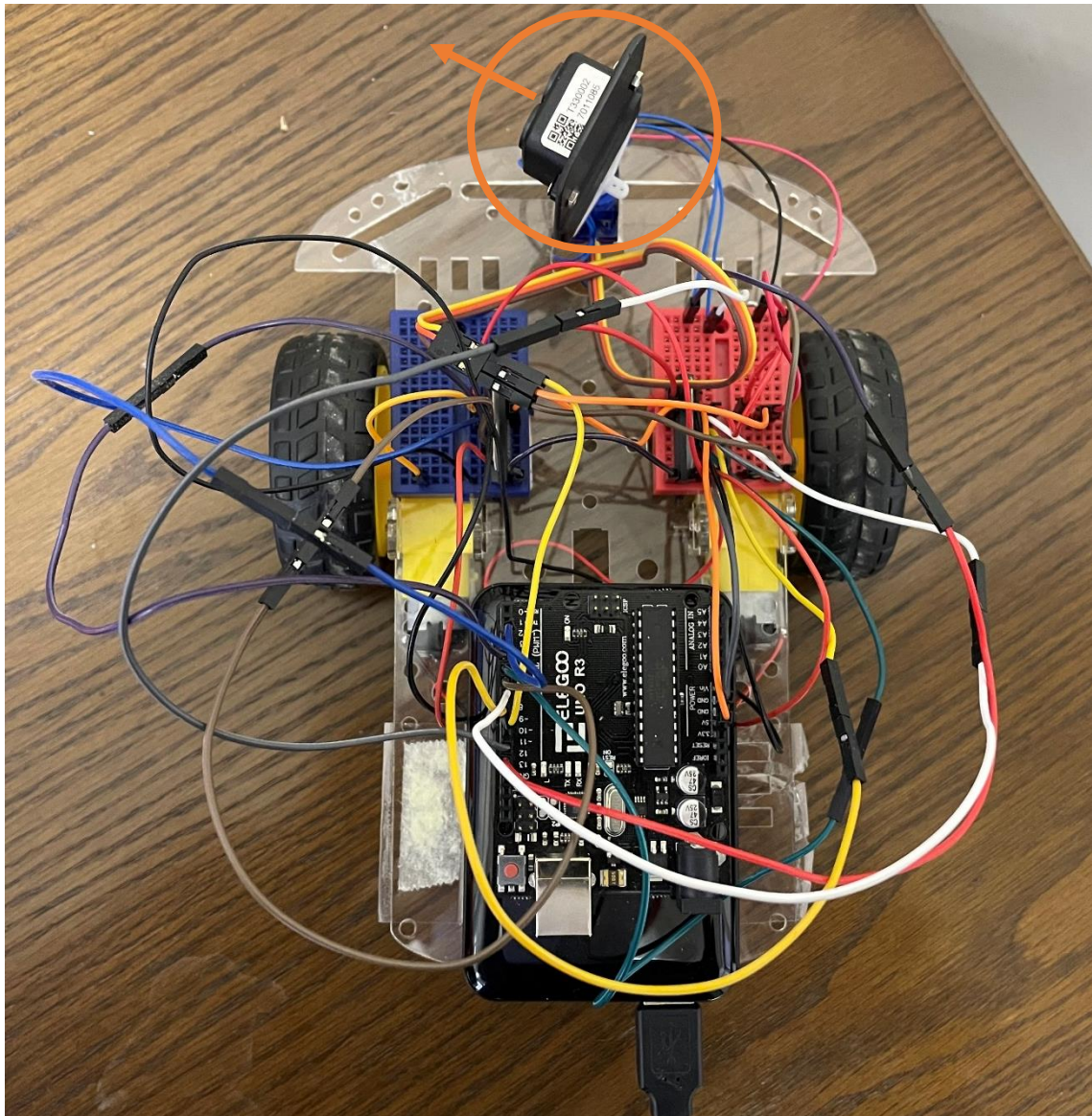


図 8. サーボモータの向きを表した図

VI. 結果と考察

図 9 に完成した自律走行カーを示す。走行結果は、12 分で 1 周を完走することに成功した。しかし、1 つ目、2 つ目、3 つ目の曲がり角まで順調に走行することが出来ていたが、最後の曲がり角を曲がった後に、壁に衝突してしまう結果となった。この原因として、曲がり角に到達した際に、車体が左に傾いていたため、ある一定時間左に傾ける処理 (`delay(80)`) を行った時に、車体が過度に曲がったことが原因ではないかと考える。また、電池残量によって、車体の速度が異なるため、`delay` の値を 80 よりも低くする必要があったのではないかと考える。

直線区間においては、概ね中心付近を走行できていたが、扉がある区間においては、壁との距離が都度異なるため、車体の傾きが大きくなる結果となった。この原因として、壁との距離が計測できる度に PWM 値の値を変更する処理を行っていたからであると考えられる。ある一定期間の距離の平均を計算し、その値によって PWM 値を変更する処理に変更することで、扉のように急に壁との距離が変わる箇所においても良好に走行が出来るのではないかと考える。

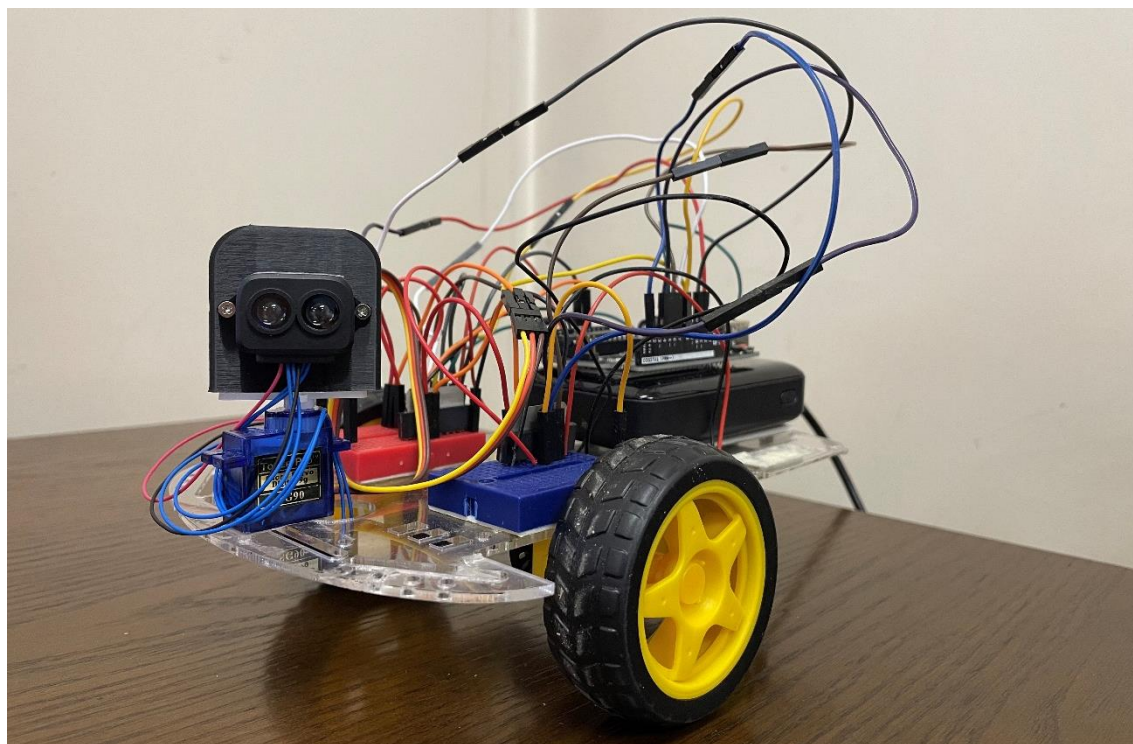


図 9. 完成した自律走行カー

VII. 感想

10 月から自律走行カー製作に取り掛かり、多少遅れはあったものの計画通りに作業を行うことが出来た。また、当初はセンサの固定角度を壁に対して垂直に固定していたこと思ったように走行せず、複数のセンサを使うことも検討したが、最終的には壁に対して斜めに固定することで 1 つのセンサのみで比較的安定して走行することが出来た。今回の講義では、日々、コンセプトとはずれていないか？マイルストーン通りに作業が進んでいるか？といったことを考え作業を行ってきたので、他の学生に比べて時間に余裕を持って行動することが出来ていたのではないかと思う。改めて、コンセプトとマイルストーンの重要性に気づかされた。

この講義で学んだことを、今後の研究や就職した際に活かしていこうと思う。

付録 1 プログラムコード

```
1  #include <SoftwareSerial.h>
2
3  // https://garchiving.com/pwm-control-with-arduino/
4  // https://qiita.com/thorikawa/items/a6377d2d4b4535dd9004
5  #include <Servo.h>
6
7  Servo penguin;
8
9
10 //曲がり角の処理
11 #define wall_distant 250 // cm
12
13 int flag_right=0;
14 unsigned long int frame_count = 0;
15
16 int left_servo_count = 0;
17 int right_servo_count = 0;
18
19
20 //サーボモータの傾き
21 #define servo_right_curve 40
22 #define servo_left_curve 147
23
24
25 //DC モータ
26 //右
27 #define PIN_RIGHT_IN1 7
28 #define PIN_RIGHT_IN2 8
29 #define PIN_RIGHT_VREF 5
30
31 //左
32 #define PIN_LEFT_IN1 4
33 #define PIN_LEFT_IN2 2
34 #define PIN_LEFT_VREF 6
35
```

36	//DC モーターのスピード
37	int left_Speed;
38	int right_Speed;
39	
40	//最大と最小のスピードを指定
41	#define High_Speed_right 195
42	#define Low_Speed_right 150
43	#define High_Speed 195
44	#define Low_Speed 150
45	
46	
47	
48	
49	//サーボモータ
50	String servo_direction; //サーボモータの向き
51	
52	
53	//LiDAR
54	SoftwareSerial Serial1(12, 11);
55	float dist; //actual distance measurements of LiDAR
56	int strength; //signal strength of LiDAR
57	float temprature;
58	int check; //save check value
59	int i;
60	int uart[9]; //save data measured by LiDAR
61	const int HEADER = 0x59; //frame header of data package
62	
63	
64	
65	
66	//前進
67	void foward(int left_speed, int right_speed)
68	{
69	digitalWrite(PIN_RIGHT_IN1, HIGH);
70	digitalWrite(PIN_RIGHT_IN2, LOW);
71	digitalWrite(PIN_LEFT_IN1, HIGH);

72	digitalWrite(PIN_LEFT_IN2, LOW);
73	analogWrite(PIN_LEFT_VREF, left_speed);
74	analogWrite(PIN_RIGHT_VREF, right_speed);
75	}
76	
77	
78	//右
79	void right(int left_speed, int right_speed)
80	{
81	digitalWrite(PIN_RIGHT_IN1, LOW);
82	digitalWrite(PIN_RIGHT_IN2, HIGH);
83	digitalWrite(PIN_LEFT_IN1, HIGH);
84	digitalWrite(PIN_LEFT_IN2, LOW);
85	analogWrite(PIN_LEFT_VREF, left_speed);
86	analogWrite(PIN_RIGHT_VREF, right_speed);
87	}
88	
89	//左
90	void left(int left_speed, int right_speed)
91	{
92	digitalWrite(PIN_RIGHT_IN1, HIGH);
93	digitalWrite(PIN_RIGHT_IN2, LOW);
94	digitalWrite(PIN_LEFT_IN1, LOW);
95	digitalWrite(PIN_LEFT_IN2, HIGH);
96	analogWrite(PIN_LEFT_VREF, left_speed);
97	analogWrite(PIN_RIGHT_VREF, right_speed);
98	}
99	
100	//後ろ
101	void back(int left_speed, int right_speed)
102	{
103	digitalWrite(PIN_RIGHT_IN1, LOW);
104	digitalWrite(PIN_RIGHT_IN2, HIGH);
105	digitalWrite(PIN_LEFT_IN1, LOW);
106	digitalWrite(PIN_LEFT_IN2, HIGH);
107	analogWrite(PIN_LEFT_VREF, left_speed);

```

108   analogWrite(PIN_RIGHT_VREF, right_speed);
109 }
110
111
112 //停止
113 int stop_(int left_speed, int right_speed)
114 {
115     digitalWrite(PIN_RIGHT_IN1, LOW);
116     digitalWrite(PIN_RIGHT_IN2, LOW);
117     digitalWrite(PIN_LEFT_IN1, LOW);
118     digitalWrite(PIN_LEFT_IN2, LOW);
119     analogWrite(PIN_LEFT_VREF, left_speed);
120     analogWrite(PIN_RIGHT_VREF, right_speed);
121 }
122
123
124
125 //LiDAR
126 void LiDAR()
127 {
128     if (Serial1.available())
129     {
130         if (Serial1.read() == HEADER)
131         {
132             uart[0] = HEADER;
133
134             if (Serial1.read() == HEADER)
135             {
136                 uart[1] = HEADER;
137
138                 for (i = 2; i < 9; i++)
139                 {
140                     uart[i] = Serial1.read();
141                 }
142
143                 check = uart[0] + uart[1] + uart[2] + uart[3] + uart[4] + uart[5] + uart[6] + uart[7];

```

144	
145	if (uart[8] == (check & 0xff))
146	{
147	dist = float(uart[2] + uart[3] * 256);
148	
149	//strength = uart[4] + uart[5] * 256;
150	
151	//temprature = uart[6] + uart[7] * 256;
152	//temprature = temprature / 8 - 256;
153	}
154	}
155	}
156	}
157	
158	if (dist > 800) dist = 800;
159	if (dist < 0) dist = 0;
160	}
161	
162	
163	
164	
165	void setup() {
166	// put your setup code here, to run once:
167	
168	penguin.attach(9);
169	
170	//タイヤ
171	pinMode(PIN_RIGHT_IN1, OUTPUT);
172	pinMode(PIN_RIGHT_IN2, OUTPUT);
173	pinMode(PIN_LEFT_IN1, OUTPUT);
174	pinMode(PIN_LEFT_IN2, OUTPUT);
175	
176	//LiDAR
177	Serial.begin(9600);
178	Serial1.begin(115200);
179	

180	
181	//サーボモータ
182	penguin.write(servo_left_curve);
183	
184	//LiDAR の向きを指定 (初期値)
185	servo_direction = "left";
186	delay(5000);
187	
188	//計測
189	LiDAR();
190	
191	}
192	
193	
194	
195	
196	void loop() {
197	
198	
199	int left_tire_R, right_tire_R; //右側の壁に沿って走行
200	int left_tire_L, right_tire_L; //左の壁に沿って走行
201	
202	
203	left_tire_R = int((float(High_Speed_right - Low_Speed_right) / 200.0)*dist +
204	Low_Speed_right);
205	right_tire_R = int((float(Low_Speed_right - High_Speed_right) / 200.0)*dist +
206	High_Speed_right);
207	
208	left_tire_L = int((float(Low_Speed - High_Speed) / 200.0)*dist + High_Speed);
209	right_tire_L = int((float(High_Speed - Low_Speed) / 200.0)*dist + Low_Speed);
210	
211	
212	
213	if(servo_direction == "left") //もし、サーボが左を向いていたら、
214	{
215	

216	if(left_servo_count == 0)
217	{
218	penguin.write(servo_left_curve);
219	}
220	
221	
222	left_servo_count++;
223	
224	if(dist >= wall_distant)
225	{
226	
227	foward(Low_Speed, High_Speed+20);
228	delay(80);
229	
230	
231	
232	if(flag_right == -1)
233	{
234	flag_right = -1;
235	
236	}
237	else
238	{
239	flag_right=1;
240	}
241	
242	
243	left_servo_count = 0;
244	
245	frame_count = 0;
246	}
247	
248	
249	if(flag_right == 1)
250	{
251	

252	if(dist<wall_distant)
253	{
254	frame_count++;
255	
256	if(frame_count == 80000) //80000 回, 壁との距離が 2.5m 以内なら,
257	{
258	stop_(0,0);
259	penguin.write(servo_right_curve);
260	delay(1000);
261	servo_direction = "right";
262	}
263	}
264	
265	}
266	
267	LiDAR();
268	foward(left_tire_L, right_tire_L); //正面に進む
269	
270	}
271	
272	
273	
274	if(servo_direction == "right") //もし, サーボが右を向いていたら,
275	{
276	
277	LiDAR();
278	foward(left_tire_R, right_tire_R); //正面に進む
279	
280	if(dist >= wall_distant) //壁との距離が遠くなったら (曲がり角)
281	{
282	
283	servo_direction = "left";
284	
285	right_servo_count = 0;
286	
287	flag_right = -1;

288	}
289	
290	
291	right_servo_count++;
292	
293	}
294	
295	}