災害探索ロボットの作成

和氣　光志　延原　凰汰

１．研究概要

近年、地震や豪雨などの災害が増加し、危険な場所に人が立ち入らずに状況を把握できるロボットの必要性が高まっている。しかし、既存の災害対応ロボットは高価で扱いが難しいという課題がある。そこで本研究では、より安価で使いやすい探索ロボットの開発を目指した。

具体的には、スマホとロボットがあれば、ロボットが発信するネットワークに接続し、指定された URL へアクセスするだけで、カメラ映像を見ながら操作できる仕組みを開発した。

２．研究の具体的内容

（１）使用部品

Seeed Studio XIAO ESP32 S3 Sense

図１　マイコン

ロボサイトギヤードモーター(60:1)

図２　モーター

L298N モータドライバ

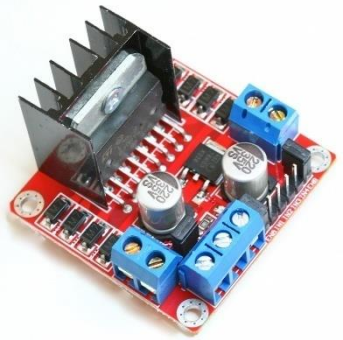

図３　モータドライバ

（２）マイコンの機能と選定理由

ESP32-S3 はカメラ機能を備え、デュアルコア CPU により映像ストリーミングと制御を同時に実行できた。また、アクセスポイントとして Wi-Fi を発信し、無線通信を行うことも可能である。これらにより、本システムで必要となる処理性能と通信機能を１つのマイコンで実現できる点から選定した。

（３）開発環境

開発には Arduino IDE を使った。（図４）
開発環境がシンプルで分かりやすい点や、実習で実際に使ったことがあり、自分たちで も理解しやすいと思ったため使用することにした。

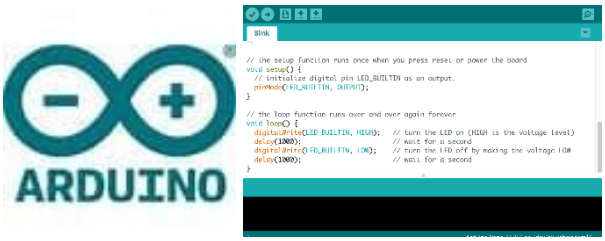

図４　Arduino IDE

（４）使用言語・ライブラリ

使用言語：HTML、CSS、C/C++、JavaScript
ライブラリ：WiFi、AsyncTCP、
esp_camera、ESPAsyncWebServer

※ライブラリとはプログラムで頻繁に使用される機能を再利用可能な形でまとめたものを指す。

（５）プログラム

映像ストリーミングと制御と操作画面をモジュールとして個別にコーディングし、修正や機能追加を行いやすいようにした。

＜各動作のプログラムの説明（抜粋）＞

１）映像ストリーミング

```
#define PWDN_GPIO_NUM    -1
#define RESET_GPIO_NUM   -1
#define XCLK_GPIO_NUM    10
#define SIOD_GPIO_NUM    40
#define SIOC_GPIO_NUM    39
#define Y9_GPIO_NUM      48
#define Y8_GPIO_NUM      11
#define Y7_GPIO_NUM      12
#define Y6_GPIO_NUM      14
#define Y5_GPIO_NUM      16
#define Y4_GPIO_NUM      18
#define Y3_GPIO_NUM      17
#define Y2_GPIO_NUM      15
#define VSYNC_GPIO_NUM   38
#define HREF_GPIO_NUM    47
#define PCLK_GPIO_NUM    13
```

このプログラムは使用するカメラピンの初期化を行っている。

```
void setupCamera() {
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;

  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
  config.frame_size = FRAMESIZE_QVGA;
  config.jpeg_quality = 12;
  config.fb_count = 2;
  config.grab_mode = CAMERA_GRAB_LATEST; // 最新フレーム優先


  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x\n", err);
    while (true) delay(100);
  } else {
    Serial.println("Camera init success");
  }
}
```

このプログラムは、マイコンに接続されたカメラモジュールの性能の設定を行うものである。

カメラのデータピン、制御ピン、クロック周波数、画像フォーマット、解像度、JPEG 品質などの各種設定を行える。画像形式は JPEG、解像度は QVGA とし、フレームバッファは 2 枚分確保することで安定した画像取得を行うこととした。

設定後、カメラの初期化を行い、初期化に失敗した場合はエラーメッセージを表示して処理を停止する。初期化が成功した場合は、シリアルモニタに成功メッセージを表示する。

※フレームバッファとは、カメラが撮影した「1 枚分の画像データ」を一時的に保存しておくメモリ領域のことである。

```
server.on("/capture", HTTP_GET, [](AsyncWebServerRequest *req){
  if (!cameraEnabled) {
    req->send(200, "text/plain", "Camera OFF");
    return;
  }
  camera_fb_t *fb = esp_camera_fb_get();
  if (!fb) {
    req->send(500, "text/plain", "Capture fail");
    return;
  }
  AsyncWebServerResponse *res = req->beginResponse_P(200, "image/jpeg", fb->buf, fb->len
  res->addHeader("Cache-Control", "no-cache, no-store, must-revalidate");
  req->send(res);
  esp_camera_fb_return(fb);
});
```

このプログラムは、カメラモジュールから画像を取得し、操作画面に送信するものである。

228 行でカメラから 1 フレームを取得し、233 行で取得した画像データを JPEG 形式の HTTP レスポンスとしてクライアントに送信する。カメラが無効の場合は撮影を行わず、フレーム取得に失敗した場合はエラーを返す。236 行で画像送信後はフレームバッファを解放し、メモリリークを防止している。

※メモリリークとは、プログラムが使用したメモリを解放せず、再利用できない状態になる現象である。メモリリークが発生し、読み込みが行われないというエラーが発生した。

２）制御

```
240  server.on("/cmd", HTTP_GET, [](AsyncWebServerRequest *req){
241    if (req->hasParam("motor")) {
242      String cmd = req->getParam("motor")->value();
243      handleMotorCommand(cmd);
244      req->send(200, "text/plain", "OK: " + cmd);
245    } else req->send(400, "text/plain", "Missing motor param");
246  });
```

```
168  void handleMotorCommand(String cmd) {
169    stopAllMotors();
170    if (cmd == "main_forward") {
171      digitalWrite(MMOTER_RIGHT1, HIGH);
172      digitalWrite(MMOTER_LEFT1, HIGH);
173    } else if (cmd == "main_backward") {
174      digitalWrite(MMOTER_RIGHT2, HIGH);
175      digitalWrite(MMOTER_LEFT2, HIGH);
176    } else if (cmd == "main_left") {
177      digitalWrite(MMOTER_LEFT1, HIGH);
178      digitalWrite(MMOTER_RIGHT2, HIGH);
179    } else if (cmd == "main_right") {
180      digitalWrite(MMOTER_RIGHT1, HIGH);
181      digitalWrite(MMOTER_LEFT2, HIGH);
182    } else if (cmd == "sub_ul") {
183      digitalWrite(SMOTER_LEFT1, HIGH);
184    } else if (cmd == "sub_up") {
185      digitalWrite(SMOTER_LEFT1, HIGH);
186      digitalWrite(SMOTER_RIGHT1, HIGH);
187    } else if (cmd == "sub_ur") {
188      digitalWrite(SMOTER_RIGHT1, HIGH);
189    } else if (cmd == "sub_dl") {
190      digitalWrite(SMOTER_LEFT2, HIGH);
191    } else if (cmd == "sub_down") {
192      digitalWrite(SMOTER_LEFT2, HIGH);
193      digitalWrite(SMOTER_RIGHT2, HIGH);
194    } else if (cmd == "sub_dr") {
195      digitalWrite(SMOTER_RIGHT2, HIGH);
196    }
197  }
```

このプログラムは、HTTP 通信を用いてモーター制御コマンドを受信し、実行するためのサーバ側処理および進行方向に合わせたモータの制御を定義した関数である。

240行で/cmdというパスに対するGETリクエストを受け付け、クエリパラメータとして送信されたモーター制御命令を処理する。リクエストにmotorパラメータが含まれている場合、その値を取得してhandleMotorCommand()関数に渡し、対応するモーター制御を実行する。制御が正常に受理された場合は、HTTPステータスコード200を返し、実行したコマンドを含むテキストレスポンスをクライアントに送信する。

一方、motor パラメータが存在しない場合は、不正なリクエストとして HTTP ステータスコード 400 を返し、必要なパラメータが不足していることを示すエラーメッセージを送信する。

３）操作画面とスクリプト

図５は実際にスマートフォンに表示される画面である。これはプロトタイプモデルで緊急停止ボタンやメインクローラのボタン配置を追加する予定である。

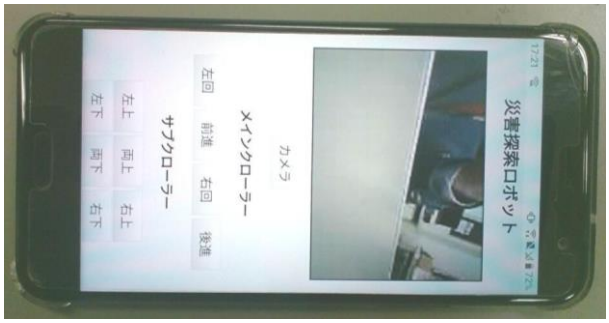

図５　操作画面

```
83   const char* htmlPage = R"rawliteral(
84   <!DOCTYPE html>
85   <html lang="ja">
86   <head>
87     <meta charset="UTF-8">
88     <meta name="viewport" content="width=device-width, initial-scale=1">
89     <title>災害探索ロボット</title>
90     <style>
91       body { text-align: center; font-family: sans-serif; background-color: #f0f0f0; }
92       h2 { margin-top: 0px; }
93       button { width: 70px; height: 40px; font-size: 18px; margin: 1px; }
94       .section { margin: 20px 0; }
95       .camera { width: 320px; height: auto; border: 2px solid #333; margin-bottom: 10px;
96     </style>
97   </head>
98   <body>
99     <h2>災害探索ロボット</h2>
100    <img id="cam" class="camera"><br>
101    <button onclick="toggleCamera()">カメラ</button>
102    <div class="section">
103      <h3>メインクローラー</h3>
104      <button onclick="send('main_left')">左回</button>
105      <button onclick="send('main_forward')">前進</button>
106      <button onclick="send('main_right')">右回</button>
107      <button onclick="send('main_backward')">後進</button>
108    </div>
109    <div class="section">
110      <h3>サブクローラー</h3>
111      <button onclick="send('sub_ul')">左上</button>
112      <button onclick="send('sub_up')">両上</button>
113      <button onclick="send('sub_ur')">右上</button><br>
114      <button onclick="send('sub_dl')">左下</button>
115      <button onclick="send('sub_down')">両下</button>
116      <button onclick="send('sub_dr')">右下</button>
117    </div>
```

このプログラムはスマートフォンに表示される操作画面を作成した。

具体的には 100 行から 101 行でカメラの映像と 102 行から 117 行で操作画面を作った。カメラの映像はカメラボタンを押さないと映像がストリーミングされないようにした。

このプログラムは、カメラの ON/OFF 制御

```
118  <script>
119    let camOn = false;
120
121    function toggleCamera() {
122      camOn = !camOn;
123      fetch('/cam?state=' + (camOn ? 'on' : 'off'));
124      if (camOn) updateCam();
125    }
126
127    function updateCam() {
128      if (!camOn) return;
129
130      const img = document.getElementById("cam");
131      const newSrc = "/capture?_=" + Date.now();
132
133      img.onload = () => {
134        setTimeout(updateCam, 120);   // 少し早めに
135      };
136
137      img.onerror = () => {
138        console.warn("Retry...");
139        setTimeout(updateCam, 200);   // エラー時は低負荷
140      };
141
142      img.src = newSrc;
143    }
144
145    function send(cmd) {
146      fetch('/cmd?motor=' + cmd)
147        .then(r => r.text()).then(console.log)
148        .catch(console.error);
149    }
150  </script>
```

と、カメラモジュールから取得した画像を操作画面に周期的に表示するため JavaScript コードである。

121 行の toggleCamera()関数ではカメラの状態を切り替え、現在の状態を HTTP リクエストとしてサーバへ送信する。124 行でカメラが有効になった場合は、127 行の updateCam()関数を呼び出して画像更新処理を開始する。

updateCam() 関数では、カメラが有効な間、一定間隔で画像を取得する。取得する画像 URL にはタイムスタンプを付与することでキャッシュを防止し、常に最新のフレームを取得できるようにしている。画像の読み込みが成功した場合は約 120ms 後に次のフレームを要求し、失敗した場合は負荷を抑えるため約 200ms 後に再試行する。

また、145 行の send() 関数はモーター制御などのコマンドを HTTP リクエストとしてサーバに送信し、その応答をコンソールに出力する役割を持つ。これにより、カメラ映像の表示とデバイス操作を同一画面から行うことが可能となっている。

※スクリプトとは、ユーザー操作や時間経過に応じてページ内容を変化させることができるプログラムコードのことである。

※カメラの ON/OFF 状態を切り替えるのはメモリの使用率を下げて、リンクへのアクセスを軽量化するためである。

（５）3DCAD について

部品の製作には 123D Design（図 6）を使った。

理由としては、オブジェクトの変形が簡単に行える点や、操作が簡単にできる点からこれを使った。

図６ 123D DESIGN

（６）車体フレームについて

本体のパーツは 3D プリンターを用いて制作した。（図７）

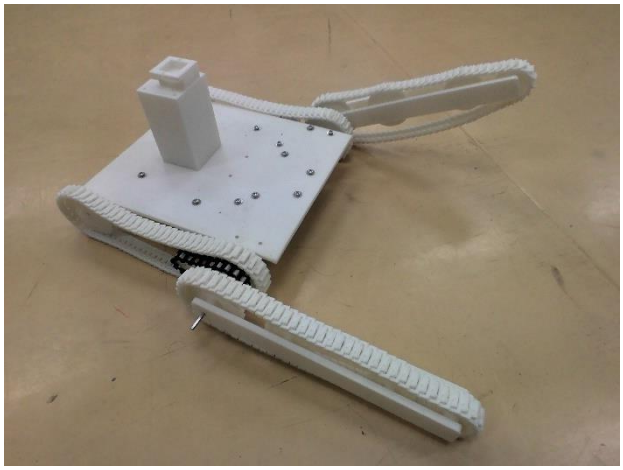

図７　プロトタイプ本体

モーター、モータドライバを取り付けられ

るように高さを 40mm、横幅を 150mm。階段を登れるように奥行を 200mm にした。また、後輪を回すための軸は、一本の長い棒にしてしまうと、左右に曲がるとき、お互いのタイヤの動きが干渉してしまうため、より短いものを一つずつ取り付けて片方ずつ動かせられるようにした。（図８）

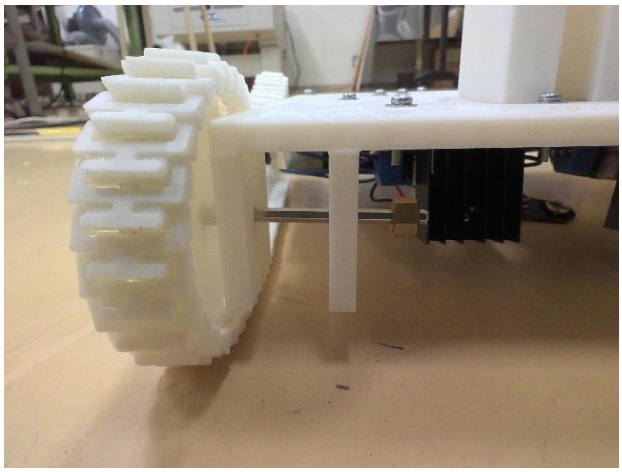

図８　後輪軸

（７）サブクローラについて

サブクローラを動かすために、サブクローラを歯車でつなげ、図９のようにもう一つのモーターの回転をサブクローラの動きに変換した。

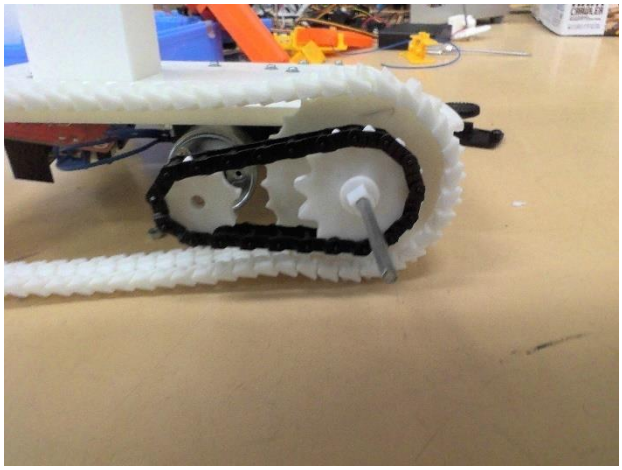

図９　サブクローラの駆動部分

大きさに関しては、階段の奥行 250mm、高さ 200mm を登れるようにフレームと歯車を合わせて全体が 200mm になるようにした。また、本体と接した部分の歯車は本体部分の歯車と同じ大きさにし、段差に乗り上げた際の負荷を少なくし、それでも全体の重量を減らせるように先端部分に行くほど細くなるようにした。（図 10）

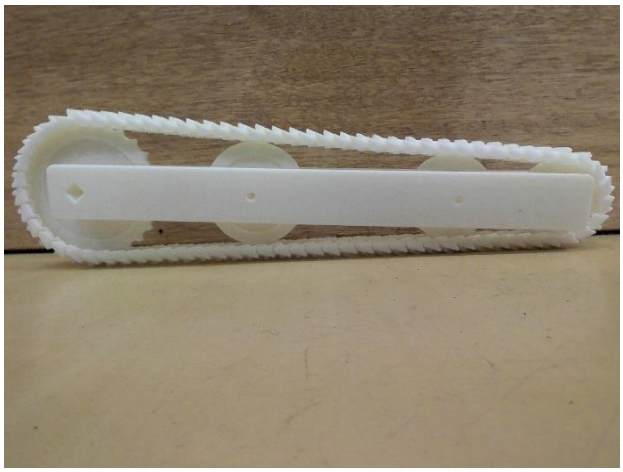

図 10　サブクローラ本体

はじめはサブクローラの長さと履帯の長さが合っておらず、うまくかみ合わなかった。そのため前後のタイヤの感覚を調節できるようにしたものを用意し、そこから得られたデータを用いて長さのかみ合ったサブクローラを作成した。さらに、段差などに乗った時に履帯が沈み込んでしまわないように中央に二つのタイヤを取り付けた。

（８）モーターについて

モーターには DAISEN のロボサイトモーターを活用した。3DCAD（図６）で制作した履帯を使用するため、モーターそのものが大きすぎると地面にこすれないようにするためにクローラの大きさをさらに上げる必要が出てきた。そのため、高さをある程度まで抑えられるこのモーターを使用した。

３．研究のまとめ

・延原風汰（ハード）

余裕をもって活動を始めたため、基本的なパーツを用意してから細かい修正に入ることができた。特に 3D プリンターを使う場合、制作には長い時間がかかり、また一回で成功する保証もないことも実感したため、より早いうちに手を付けることの大切さを実感しました。そして、簡単なソフトを使ったとはいえ、これまで 3DCAD を使ったことはなかったため、はじめは、オブジェクトを過剰に増やしてしまう、円の中心があっていないなどのミスがあった。それでも、ミスをした理由を考え、ほかの人に助けを求めるなどして、より正確に作れるようになった。この経験から、はじめは苦手意識を持ったものでも、正しい知識と経験を積むことで新しいスキルを身に着けられると知れた。

・和氣光志（ソフト）

課題研究を通して、ものを一から自分たちで作ることの大変さを深く実感した。これまでは、与えられた課題に沿って回路の作成やプログラミングを行ってきたが、今回は実習のように先生につきっきりで教えてもらうのではなく自分で調べ、考えながら研究を進める必要があった。そのため、これまでに経験したことのない作業が多く、非常に苦労した。

通信方法については、当初は同期通信を用いてプログラミングを行っていた。しかし、カメラのストリーミング処理と制御処理を統合した際に、両者を同時に実行できないという問題が発生した。非同期通信を用いたプログラミングの経験がなかったため、原因の特定に時間を要したが、試行錯誤を重ねることで問題を解決することができた。

また、参考となる文献や情報が少なく、情報収集にも苦労したが、毎日放課後に残って作業を続けることで、研究をある程度形にすることができた。この経験を通して、ものを一から作り上げる力が身につき、大きな達成感を得ることができた。さらに、今後新たに何かを一から制作する際にも、自分たちにはそれを実現できる力があるという自信につながった。

参考文献

ESP32　初歩から応用まで：　WiFi　Car　の製作 Kindle 版
https://amzn.asia/d/3kKi01S
ケンじーの備忘録チャンネル
https://www.youtube.com/@kenjii70
CSS 入門：基本の使い方を徹底解説
https://josysnavi.jp/2025/basic-css-lecture
Gear Model For 3D Printer
https://knowhave.main.jp/gear/indexjp.php
3D プリンターによるキャタピラ(無限軌道)の作成
https://fabble.cc/ytani01/3dprintertrackbelt