

金沢工業大学

工学部

トイレ掃除ロボット Happy Burger の開発

プロジェクトデザインⅢ

プロジェクトレポート

所属 口ボティクス学科
学籍番号 1512539
クラス番号 4ER1-71
氏名 湯田 晴也
指導教員 出村 公成

目次

第1章 はじめに	2
第2章 掃除とロボット	3
2.1 移動型ロボットによる掃除	3
2.2 設置型ロボットによる掃除	5
2.3 従来研究における課題	5
第3章 トイレ掃除ロボット Happy Burger の開発	7
3.1 要求仕様	7
3.2 技術的課題と解決方針	7
3.3 提案するアイデア概要	8
3.3.1 ロボットアーム	10
3.3.2 電源系	13
3.3.3 外装	15
3.3.4 ソフトウェアシステム	16
第4章 実験	19
4.1 実験条件	19
4.1.1 便器側面の実験	22
4.1.2 トイレ床面の実験	22
4.1.3 便器上面の実験	22
4.2 実験結果	26
4.2.1 便器側面の実験	26
4.2.2 トイレ床面の実験	26
4.2.3 便器上面の実験	28
4.3 考察	30
第5章 おわりに	32
謝辞	33

参考文献..... 34

付録..... 35

第1章 はじめに

現在、老年者1人の生活は生産年齢者2.3人によって支えられている。今後も出生率が低迷し続けると仮定した場合、2065年では老年者1人は生産年齢者1.2人によって支えられることになる。そのため、生産年齢者1人が老年者を養うために抱える負担は、現在の生産年齢者1.9人が抱えている負担と等しく、論理的に述べれば、老年者1人のために割かれる生産年齢者1人当たりの時間は約半分になると言える。この超高齢社会では老年者の生活を支えることが生産年齢者の課題の一つである。そこで生産年齢者が行う生活支援活動をロボットで行なうことが求められる。

すでに案内や、治療、食事といった場面で研究が進められており、多くの生活支援の場でロボットを活用する動きがある[1]~[3]。

人が生活する空間には掃除が必要である。もちろん人が生活するうえでトイレ掃除も必要であり、同時に、トイレ掃除をロボット化する要求がある。

トイレを汚す大きな原因は便である。それ以外にもトイレ部屋が汚れる要因はいくつかある。まず、備え付けのトイレットペーパーである。トイレットペーパーを切り離す際に小さく分かれた片が紙くずとして落ち、ゴミになる。また、外履きで入れるトイレであれば砂も入る。ほかにもガムや髪の毛などトイレ掃除として対処する汚れは幅広い。本プロジェクトではトイレ掃除を、トイレットペーパーの塵取りと便器の拭き掃除と定義し、トイレを掃除できるロボットの開発を目的とする。

トイレ掃除を行うロボットの種類として、移動型ロボットや設置型ロボットが挙げられる。設置型ロボットの場合では、便器をロボット化する方法やトイレ部屋をロボット化する方法が考えられるが、ロボットはトイレ掃除を専門に行なうため、他の場面で応用が利かない。その点、移動型ロボットの場合では環境センサによる安全確認が欠かせないが、トイレ以外の場所で生活支援活動を行うことも可能である。本プロジェクトではトイレ以外のリビングなど、トイレに限定されない生活空間でのロボットによる生活支援の要求にこたえることを狙い、自律移動型の小型ロボット Happy Burger を開発した。

本レポートでは、まず2章で掃除ロボットの開発が行われた先行研究とその課題について考察する。次に3章で課題の解決方針と要求仕様を述べ、実際に開発した掃除ロボット Happy Burger を提案し、その仕様について述べる。続いて4章にその掃除能力の評価実験とその考察を述べ、5章でまとめを行う。

第2章 掃除とロボット

2.1 移動型ロボットによる掃除

薩見らの研究では乾式掃除、湿式掃除の両方を行うロボットを開発した[4]。開発されたロボットを Fig. 2.1 に示す。トイレットペーパーや砂などの落ちているものは乾式掃除で除去し、こびりついた汚れは湿式掃除で除去を行う。乾式掃除は水を使わない掃除方法であり、湿式掃除は水を使う掃除方法である。ロボット底面に取り付けられた乾式、湿式のための掃除道具を使いながら磁気テープを使用した自律移動を行い、トイレ部屋床面の掃除を行う。大きさは 500mm × 450mm × 800mm である。評価実験は掛川実験センター（パーキングエリア）のトイレで行われた。

乾式掃除にはバキュームを用いられた。バキュームは吸引式の掃除機である。湿式掃除のことを考慮し、耐水性のごみ回収パックが使用された。

湿式掃除はモップとサイドパッドで行われた。どちらもマイクロファイバ製で、ロボット底面に取り付けられており、掃除の際はアルコールが自動供給されて湿った状態を保つ。また、パッドは移動時にロボット内側に収納され、地面から浮いた位置を保つ。これには2つの理由がある。1つが衛生を保つためであり、もう1つがパッドを傷つけないためである。

青山らが研究で開発したロボットは2種類の掃除装置を取り付けることで2種類の掃除を行う[5]。1つはスイーパ式でもう1つはスクラバ式である。開発されたロボットを Fig. 2.2 に示す。スイーパ式掃除装置は、ごみを吸引して除去する乾式掃除である。スクラバ式掃除は洗浄液を散布し、ブラシで床を洗浄する湿式掃除である。ロボットは生産性、保全性についても考慮されて設計された。光ファイバジャイロを使用した自律移動を行い、床面の掃除を行う。ジャイロセンサが時間経過でどの程度誤差を蓄積するかについて評価が行われたが、清掃能力の評価は行われなかった。

乾式掃除の掃除装置としてごみ吸引部がロボット底面に設けられており、ゴミ回収タンクがロボットに搭載されている。手持ちの掃除機の使用もできるよう、ゴミ回収タンクには仕切弁が設けられている。

湿式掃除の掃除装置として水の散布機構とブラシ、スクイージーが設けられている。スクイージーは水を回収するためのゴムである。スクイージーはロボットの旋回にも対応するため、円弧状である。ロボット内のタンクから洗浄液が供給



Fig. 2.1 Toilet cleaning robot (薩見[4]から転載)

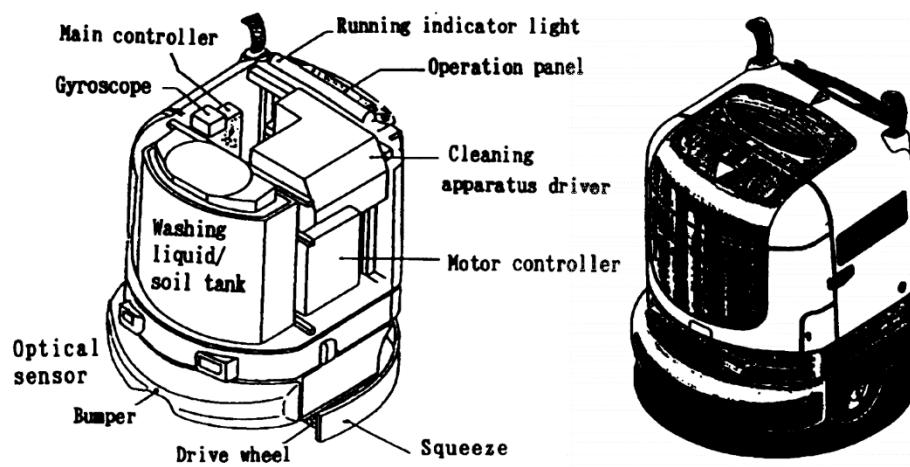


Fig. 2.2 Floor cleaning robot (Left one is scrubber-type, Right side one is sweeper-type.
青山[5]より転載)

され、回収された汚水は洗浄液と弁で仕切られ、同じタンクに戻される。

掃除環境をロボットに与える手段に地図は用いられず、教示時のロボットの位置を一つの頂点とする矩形をロボットに与える方法を取った。ロボットはその矩形を格子状に切り、通っていない升を優先して通るよう走行計画を行い、ジャイロセンサを用いて走行する。

Miyake らはガラス上を移動するロボットを提案した[6]。ビルの窓ガラスを掃除するロボットとして提案された。吸盤とタイヤで垂直な平面に付いて移動する。検証実験はガラスを使った窓ガラスモデルで行われ、カメラでロボットを撮影し、ロボットが通ったか通っていないかの視点で検証が行われた。

2.2 設置型ロボットによる掃除

Anh らが提案した手法は便座にナイロンシートを巻き付けるアイデアである[7]。ナイロンシートを自動的に巻き取ることで清潔に保つ。

Ben-Amram らはトイレのフタとしてトイレにつける形状の掃除ロボットを提案した[8]。提案されたロボットを Fig. 2.3 に示す。トイレのフタが閉められた状態でロボットは動作し、便器の中を洗浄する。

2.3 従来研究における課題

移動型のロボットの掃除装置はいずれもロボットの底面に取り付けられてきた。移動型のロボットでは掃除範囲が床などロボットの移動面に限られていて、



Fig. 2.3 Self-cleaning toilet (Ben-Amram[8]より転載)

便器の掃除を行う手法は提案されていない。また、設置型のロボットでも便器の外側を掃除する手法は提案されていない。トイレ掃除のロボット化に向け、便器をはじめ立体的なものの掃除を行える掃除装置を開発する課題がある。

ロボットの実証実験が行われた例はあったが、掃除能力についての定量的な評価は行われた例はなかった。定量的に環境を汚し、ロボットがそれを掃除する様子からロボットの掃除能力を評価する必要がある。

いずれのロボットも掃除を専門に行うように設計され、生活支援の場での活用が期待されるロボットの開発は行われなかつた。掃除以外にも生活支援の場で活躍が期待できる汎用的なロボットの開発が課題である。

第3章 トイレ掃除ロボット Happy Burger の開発

3.1 要求仕様

トイレ部屋の定義をトイレの部屋全体にすると部屋の換気扇など対応すべき範囲が広くなりすぎてロボットの構成が大きくなりすぎてしまう。そこで掃除対象であるトイレ部屋を、洋式トイレの便座までの高さまで限定した。またトイレに存在するごみにはトイレットペーパー以外にも髪の毛、さらには床面にこびりつき湿式清掃を必要とするものもあるが、今回は乾式清掃を必要とするトイレットペーパーに限定した。ロボットは塵取り掃除を行い、尿の拭き掃除を行う必要がある。また、対象とするトイレは家庭用サイズのトイレではなく掛川実験センター（パーキングエリア）サイズのトイレとした。

そこで要求仕様を以下のように定義した。

- (1) トイレに散らばった 5mm 角のトイレットペーパーをロボットが自身のごみ箱に収納できる。
- (2) トイレ側面や便座下の上面につく飛沫の 8割を除去できる。
- (3) 直径 450mm に収まる。

3.2 技術的課題と解決方針

トイレットペーパーをとるためにバキューム掃除を行う。バキューム掃除には iRobot の Roomba を使うことにした[9]。Roomba は 10 年以上前から販売され、消費者も多く信頼できる掃除機である。

トイレ掃除を行うロボットには掃除装置が必要である。薩見ら[4]の手法では床面の掃除しか行われなかつたため、掃除装置は移動機構と同じ面につけられた。しかし、便器を掃除するには掃除装置が 3 次元的に動く必要があり、土台として使用する Roomba だけでは便器を掃除できない。さらに、便器の形状が曲線であるため、掃除装置は曲線に沿うよう動かす必要がある。曲面に沿った掃除が必要であることと、生活空間でも応用されることを期待し、ロボットアームのエンドエフェクタにも取り付けることにした。エンドエフェクタに取り

付けた掃除装置で便器についていた尿の拭き掃除と床についていた尿の拭き掃除のどちらも行う。

また、便器の掃除の際、便器についていた汚れを落とすため、便器に掃除装置を押し付けて拭き掃除を行う必要がある。そのため、便器の拭き掃除には電流値制御でロボットアームを動かす方法を使用した。電流値制御はロボットアームの各関節のモータのトルクを決められるモータの制御方式である。

またロボットがトイレ部屋の中で掃除ができる必要があるため、大きさを直径 450mm に収まるとした。薩見らの開発したロボットの大きさは約 450mm × 500mm で、掛川実験センターのトイレで実証実験された前例があり、掃除ができることが証明されているため、それより小さければ実際のトイレで掃除ができるといえる。

さらに、自動ロボットとしてトイレの部屋におけるトイレとロボットの位置関係を把握する必要があるため、2DLidar と RGB-D Camera をそれぞれ自己位置推定とトイレの形状推定のために搭載することとした。RGB-D Camera はロボットアームに取り付けることで、任意の方向を見ることができる。

3.3 提案するアイデア概要

ベースに掃除ロボット、ロボットアームにはサーボモータを 5 つ用い、センサには 2DLidar と RGB-D Camera を用いた。仕様を Table 3.1 に示す。また、ロボットの外観と構成図を Fig. 3.1、Fig. 3.2 に示す。

Table 3.1 Specification

Vehicle dimension	Approx. 400 mm×350 mm×240 mm Body only
Weight	Approx. 13 kg including computer
Actuators	DC12V Dynamixel XM430-W350-R
Robotic Arm	Mikata Arm
Power Source	Two 6.6V Li-Fe Batteries
Sensors	HOKUYO 2DLidar UTM30LX, INTEL REALSENSE DEPTH CAMERA D435
Computer	ALIENWARE 13 GAMING Laptop



Fig. 3.1 Exterior of the developed robot Happy Burger

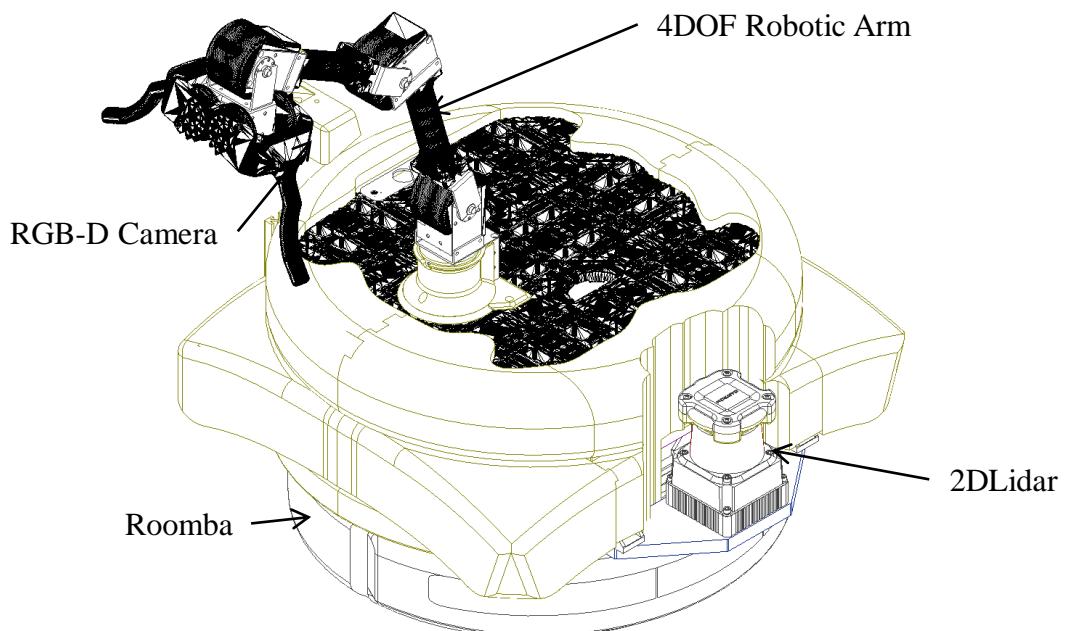


Fig. 3.2 Toilet cleaning robot Happy Burger

3.3.1 ロボットアーム

ロボットアームはトルクモードで制御を行うため、モータは電流値制御でオペレーションができるものを採用した。また、ロボットアームはエンドエフェクタをトイレに押し付けて掃除をする必要があったため、モータの必要トルクを Fig. 3.3 に示す環境で静力学の解析を行った上で決めた。2自由度ロボットアームがトイレ側面を拭く場合を考えた。ロボットアームの各リンクの長さを 150mm とし、ロボットアームのベースジョイントとトイレ側面との距離が 150mm であるとした。エンドエフェクタをトイレ側面に 10N で押し付けた場合の必要トルクをエンドエフェクタの高さがベースジョイントから y 軸方向 ± 150 mm の範囲で計算した。簡単のためロボットアーム、エンドエフェクタには体積がなく、トイレ側面は垂直であると仮定した。

2自由度ロボットアームのヤコビ行列 J は次の式で与えられる。

$$J = \begin{pmatrix} -l_1 \sin \theta_1 - l_2 \sin(\theta_1 + \theta_2) & -l_2 \sin(\theta_1 + \theta_2) \\ l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2) & l_2 \cos(\theta_1 + \theta_2) \end{pmatrix} \quad (3.1)$$

但し、 l_1 、 l_2 はリンク 1、リンク 2 の長さ、 θ_1 、 θ_2 はショルダージョイント、エルボージョイントの角度である。

また、静力学の関係式より以下の式が得られる。

$$\mathbb{T} = J^t \mathbb{F} \quad (3.2)$$

但し、 $\mathbb{T} = (T_1 \ T_2)^t$ で、 T_1 、 T_2 はそれぞれショルダージョイント、エルボージョイントにかかるトルクであり、 $\mathbb{F} = (F_x \ F_y)^t$ で、 F_x 、 F_y はそれぞれエンドエフェクタが x 軸、 y 軸に与える力である。

式 1、式 2 を用いて必要トルクを計算した。

解析結果を Fig. 3.4 に示す。安全係数を 2 としたとき、最高必要トルクは 3.00 Nm だった。XM-430-W350-R はストールトルクが 4.1Nm で必要トルクを満たしており、電流値制御が行えるモータであるためこれを採用した。

XM-430-W350-R を使用するロボットアームに Mikata Arm というオープンハードウェアがある。Mikata Arm は Fig. 3.5 に示すような 4自由度ロボットアームであり、それをロボットアームに採用した。

ベースジョイントはトイレ上面の掃除の際、端まで拭き掃除を行うときに使った。また、ベースジョイントがロボット座標系で z 軸を中心回転することで、ロボットアームにつけられたカメラをロボットの全方位を見渡すためのカメラとしても使用できる。ショルダージョイント、エルボージョイントは曲面に沿ってエンドエフェクタを動かすために使った。ショルダージョイント、エルボージョイントの 2 つがあるために、ロボットが台車を動かさなくても便器

側面の曲面に沿うようにエンドエフェクタを動かせる。ジョイント 4 はエンドエフェクタを掃除する面に向けるために使った。

ロボットのエンドエフェクタが床に届くよう、ロボットアームはロボットの中心ではなく、ロボットの右手側に寄せた。

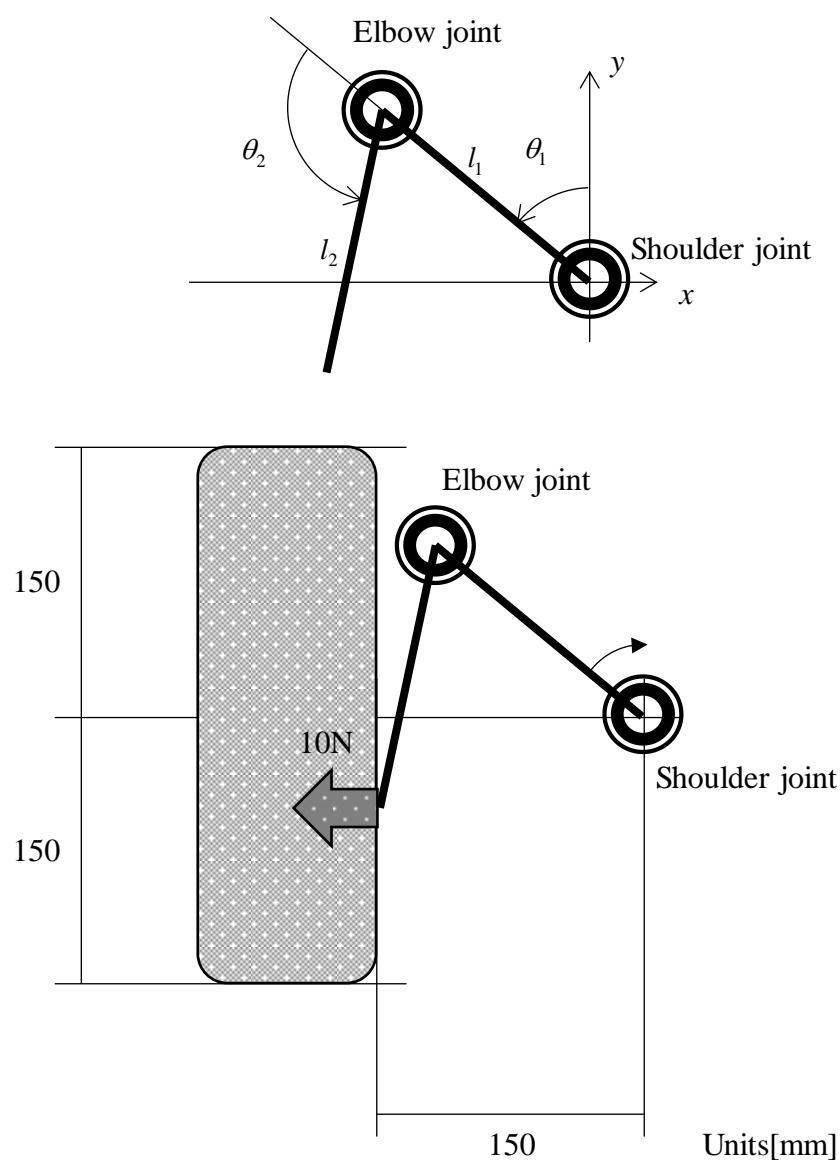


Fig. 3.3 Environment of needed torque calculated

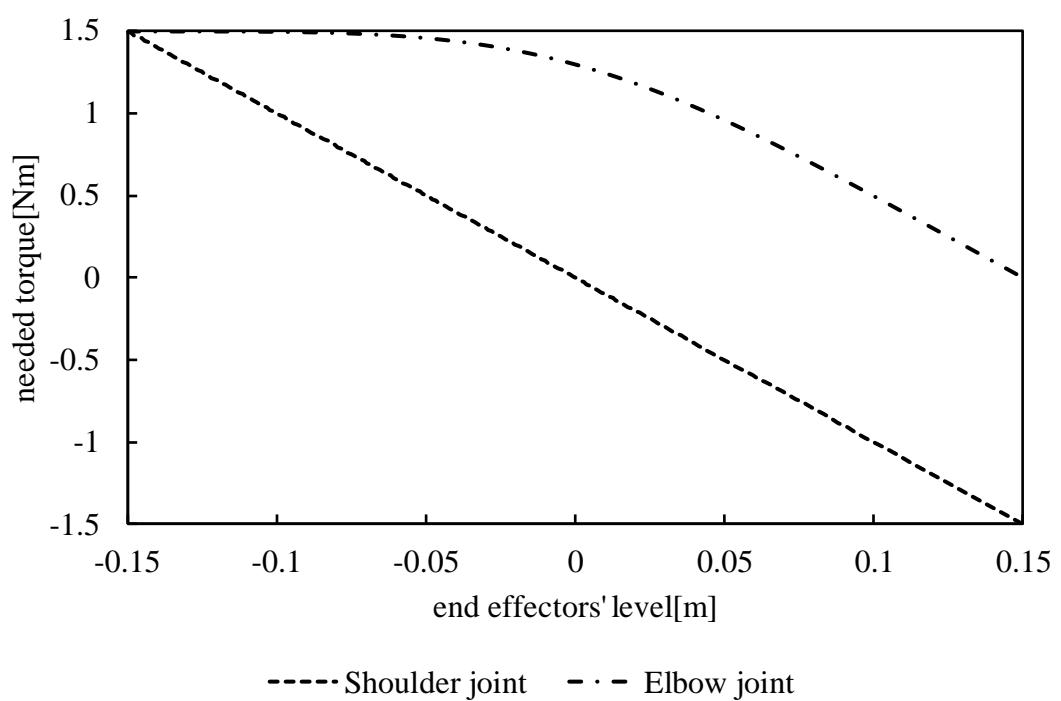


Fig. 3.4 Calculated needed motors' torque

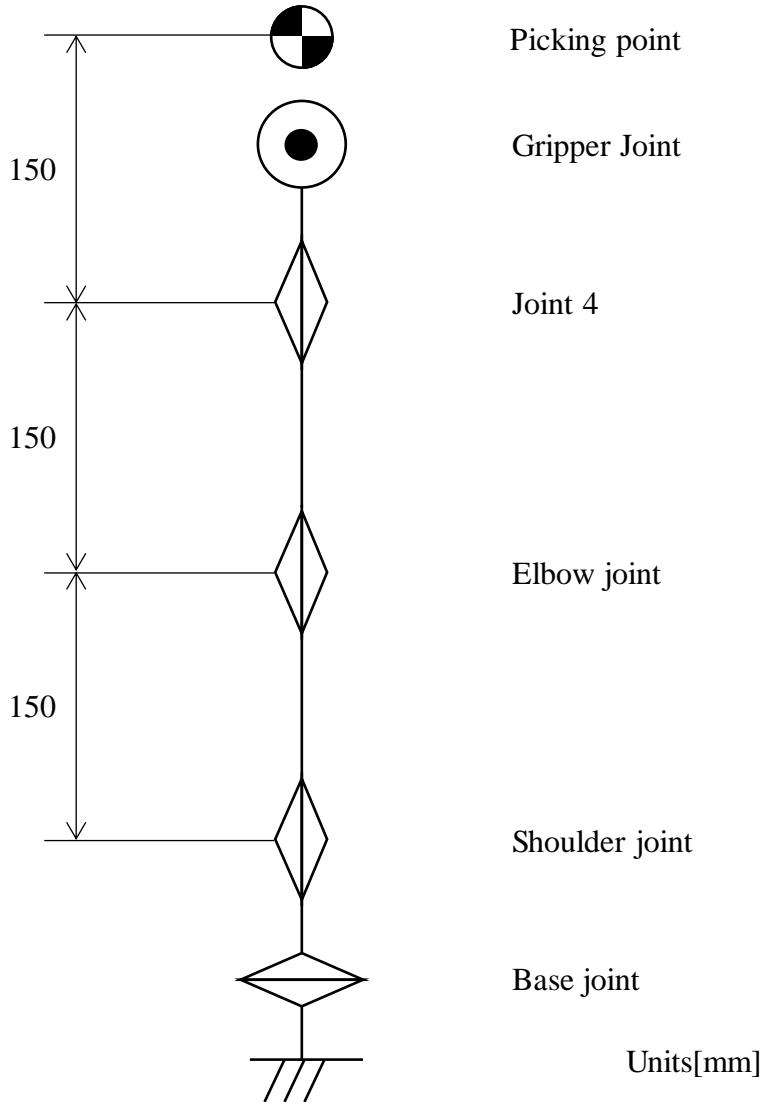


Fig. 3.5 Robotic arm link parameters

3.3.2 電源系

ロボットに搭載した電源について述べる。搭載したパーツで電源を必要とするものとその電源の供給方法についての表を table 3.2 に示す。Roomba の電源は Roomba に搭載されるバッテリーで賄った。ALIENWARE と RGB-D Camera はコンピュータに搭載されるバッテリーで賄い、2DLidar とロボットアームの電源は 6.6V のバッテリーを 2 つを直列につないだもので賄った。

回路の配線図を Fig. 3.6 に示す。Roomba、RGB-D Camera、Roomba、ロボットアーム、2DLidar を 1 つのコンピュータにつなぎ、1 つのコンピュータでロボット全体の制御を行う構成にした。

Table 3.2 Supply sources

Parts	Supply
Computer	Computer battery
Depth Camera	Computer battery
Roomba	Roomba battery
2DLidar	6.6V batteries
Robotic Arm	6.6V batteries

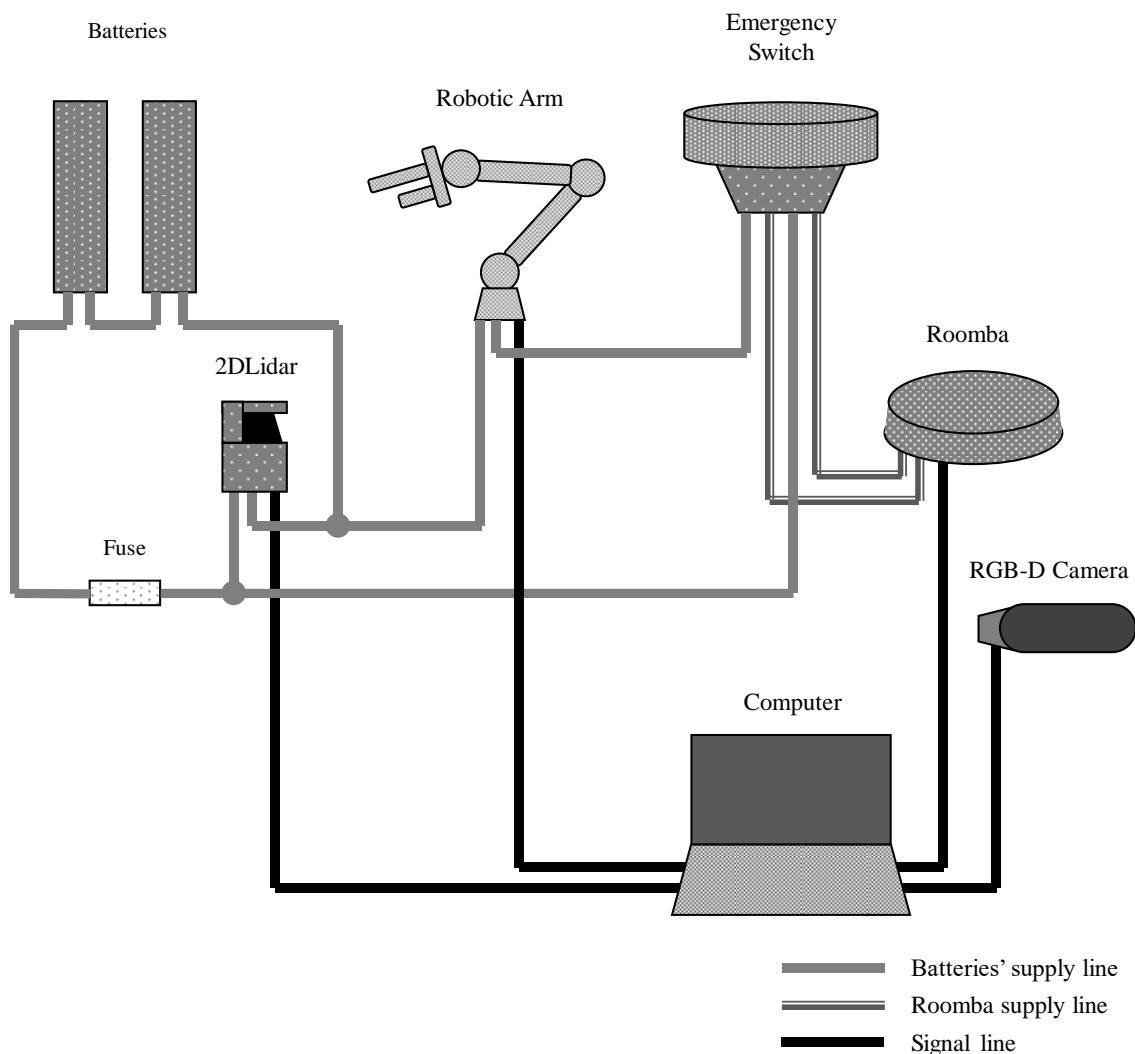


Fig. 3.6 Wiring diagram

2DLidar とロボットアームで使用する電流について解析した。計算した結果を Table 3.3 に示す。2DLidar とロボットアームは 12V で動作する。2DLidar に流れる電流は、0.7A である。また、モータの必要トルクの解析の結果、便器側面の掃除の際にモータにかかるトルクの最大値はショルダージョイント、エルボージョイントとともに 1.50Nm だった。モータの仕様上、1.50Nm のトルクがかかっている場合モータに流れる電流は 0.9A である。ここでワーストケースを、2DLidar とロボットアームを同時に使用していくて、ロボットアームのショルダージョイント、エルボージョイントとともに 1.50Nm のトルクがかかり、ジョイント 4 にも 1.50Nm のトルクがかかっている状態とする。この場合、ロボットアームだけで 2.70A の電流が流れ、合計で 3.40A の電流が流れる。すべて 12V で動作するので 40.8W の電力消費がある。30 分の使用を考え、安全率を 2 とした場合、必要電力量は 40.8Wh である。6.6V、3.2Ah のバッテリー 2 つの電力量は 42.2Wh であり、必要電力量を満たすため、これをバッテリーに選んだ。

3.3.3 外装

ロボットはミスミのアルミフレームを使った構造体内にコンピュータやバッテリー、コードを収納する場所がある。コンピュータやバッテリーなどが外に露出しないよう、外装を取り付けた。外装は 3D プリンタを用いて PLA の樹脂で製作した。生活支援の場でも溶け込めるよう、ハンバーガーを模した丸みを帯びたデザインにした。コンピュータの出し入れの際、外装を外しやすくするため、ロボットアームと反対側の、ロボット左手側の外装はそれと接触する外装に固定されないようにした。また、ロボットが障害物に衝突した際に、コンピュータを守るため、外装とコンピュータには 4mm の隙間が空くよう設計した。

Table 3.3 Calculation of electric energy

Parts	Electric Energy[Wh]
2DLidar	8.40
Shoulder joint	10.8
Elbow joint	10.8
Joint 4	10.8
Total	40.8
Batteries	42.2

3.3.4 ソフトウェアシステム

ロボットは ROS のシステムを使って動作させた。ROS のノードのつながりを Fig. 3.7 に示す。

Rqt graph を載せた。楕円で囲まれたものがノードで ROS の実行プログラムで、ROS では四角で囲まれたトピックを使ってほかのノードとやり取りをする。`/bringup` ノードは Mikata Arm を動かすためのノードで、`/ca_driver` ノードは Roomba を動かすためのノードである。Roomba と Mikata Arm との通信をそれらのノードで行い、デッドレコニングを実装した。実装したデッドレコニングの評価は行わなかった。ロボットの URDF モデルも製作した。製作したモデルを Fig. 3.8 に示す。

URDF のモデルは GAZEBO と ROS が対応しているため、実機を動かさなくとも、シミュレーター上でロボットを動かせる。シミュレーターでトイレを再現することで、ロボットがそこでどのように動作するかを検証できた。



Fig. 3.7 Rqt graph of Happy Burger



Fig. 3.8 URDF robot model for simulator

第4章 実験

4.1 実験条件

トイレの部屋モデルを Fig. 4.1 に示す構成で用意した。また、実験の際には、ごみを模した 5mm の長さに切られたトイレットペーパーをランダムな位置に置き、模擬尿で汚し、掃除が必要な状態を再現した。

トイレ部屋モデルは実際のトイレ環境で特に汚れると想定される部分を黒色にした。実験では、ゴミや尿モデルは黒色の部分に設置し、ロボットがそれを取り除くことができるか評価を行った。さらに実験を問題の単純化のために要素ごとに分け、以下 3 点で検証を行った。

(1) 便器とロボットの位置関係が取得できるかの検証

自己位置推定のために用いられる 2DLidar がどの範囲で環境値を取得できるか検証した。

(2) 5mm 角のトイレットペーパーを取り除けるかの検証

ロボットを無線操作し、無作為に置かれた 5mm 角のトイレットペーパー 5 つをいくつ取り除けるか検証した。

(3) 模擬尿として用意する、水性ペンで描かれた線を取り除けるかの検証

ロボットを無線操作し、尿を模してつけられた水性ペンの汚れをどの程度取り除けるか検証した。水性ペンの汚れは Fig. 4.2 のように付けた。

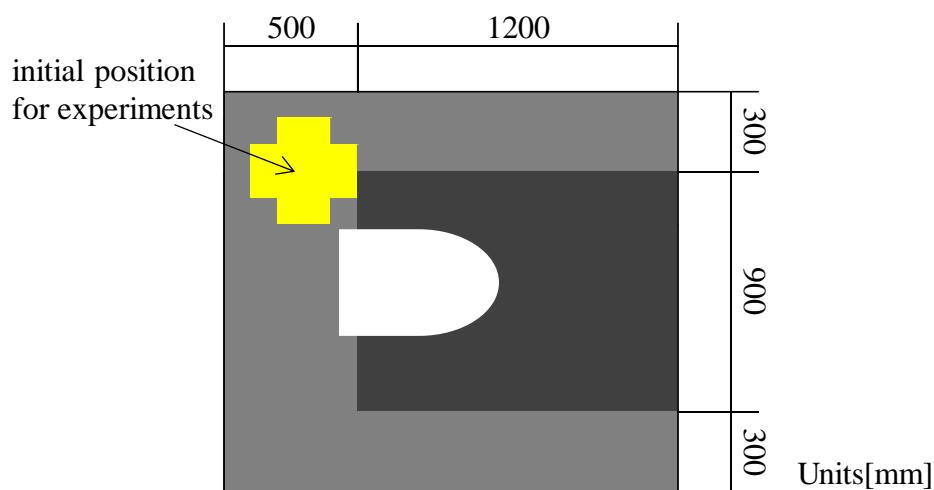


Fig. 4.1 Experiment environment

また、実験の際には便器側面の掃除の結果を Fig. 4.3、トイレ床面の掃除の結果を Fig. 4.4、便器上面の掃除の結果を Fig. 4.5 に示す環境で撮影して記録に残した。

便器側面の掃除の実験では、便器正面で便器から 150mm 離れた場所にカメラを設置した。床からの高さは 44.5mm で、ちょうど便器の中央の位置から撮影した。

トイレ床面の掃除の実験は、床からの高さ 89mm で、便器の高さと同じ高さにカメラを設置した。便器の縁から 75mm 離した。それは床面に 150mm の線を汚れとして引いたためである。

便器上面の掃除の実験では、床からの高さ 1100mm で、便器のフタが固定されている位置の真上にカメラを 10° 便器側に傾けた場所にカメラを設置した。カメラを傾けたのはカメラの画角に便器全体を移すためである。



Fig. 4.2 How to make bathroom dirty

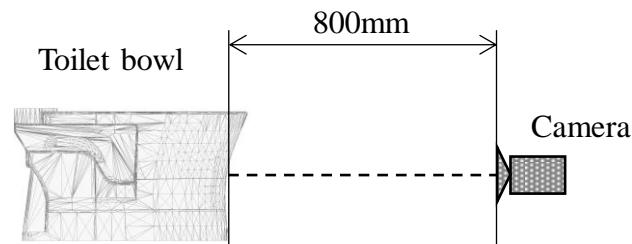


Fig. 4.3 Environment to take pictures of cleaning side of toilet bowl experiments

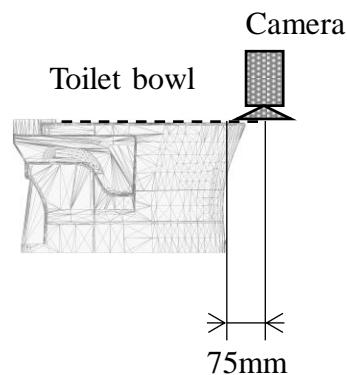


Fig. 4.4 Environment to take pictures of cleaning floor experiments

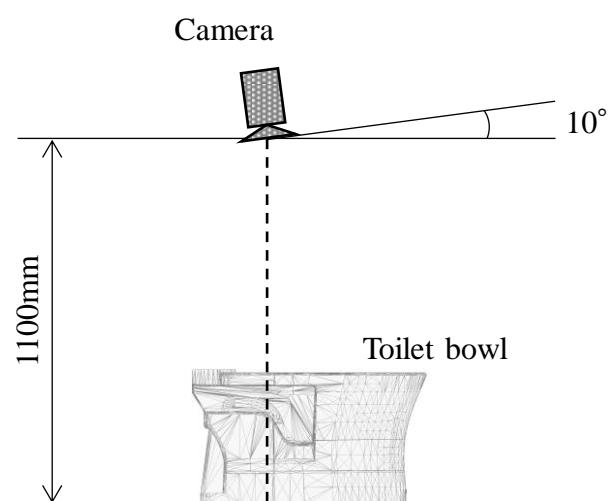


Fig. 4.5 Environment to take pictures of cleaning top of toilet bowl experiments

4.1.1 便器側面の実験

便器側面の掃除の実験時のフローチャートを Fig. 4.6 に示す。ロボットをトイレ内に置いた状態から実験を始めた。掃除方法は 2 種類ある。1 つは簡単な方法である。もう 1 つは簡単な方法では落としきれなかった汚れも落とす方法である。

まず、簡単な方法による掃除は 5 段階に分けられる。ロボットの初期位置は Fig. 11 の黄色の印である。そこから便器の正面まで便器と距離を約 20mm あけてトイレ正面まで移動し、停止した。便器の正面ではロボットの右手側が便器に面する方向であり、ロボットアームが搭載されている側でもある。ロボットアームのエルボージョイントをトルクモードにしてエンドエフェクタを便器に押し付け、ショルダージョイントを回転させてエンドエフェクタを便器の上から下まで掃除した。その際、モータは 0.35 Nm から 1.40 Nm のトルクをかけた。その後、ロボットアームをレストポジションに移動してロボット自身もまた便器との距離を 20mm あけてトイレ後方まで移動した。ここまで動作で掃除 1 回と数えた。

改善された方法による掃除は 3 段階に分けられる。初期位置から、エルボージョイントを電流値制御にして便器の下側にエンドエフェクタを押し付け、便器との距離を 200mm に保ったまま便器正面を通り、トイレ後方まで移動した。

4.1.2 トイレ床面の実験

トイレ床面の掃除の実験時のフローチャートを Fig. 4.7 に示す。ロボットをトイレ内に置いた状態から実験を始めた。エンドエフェクタを床面にあて、便器に沿って移動した。その際、モータは 0.35 Nm から 1.40 Nm のトルクをかけた。2 度同じ動きを繰り返し、それを掃除 1 回と数え、記録に残した。

4.1.3 便器上面の実験

便器上面の掃除の実験時のフローチャートを Fig. 4.8 に示す。ロボットをトイレ内に置いた状態から実験を始めた。エンドエフェクタを便器上面にあて、便器との幅を 200mm に保つように移動した。その際、モータは 0.35 Nm から 1.40 Nm のトルクをかけた。トイレ反対側まで付いたらロボットアームをレストポジションに戻した。ここまで動作で掃除 1 回と数えた。

Improved process

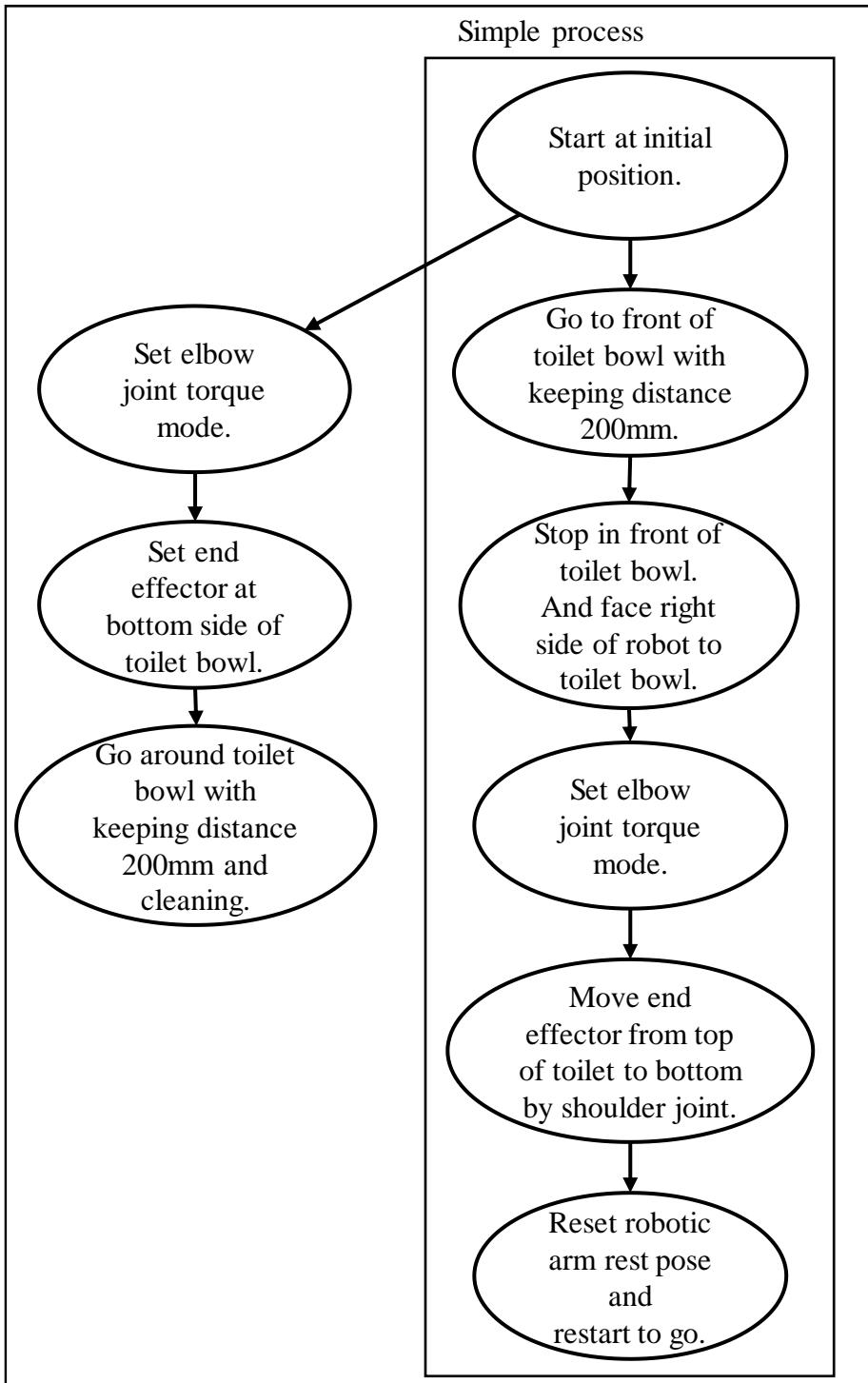


Fig. 4.6 Flow chart for experiment of cleaning side of toilet bowl experiment

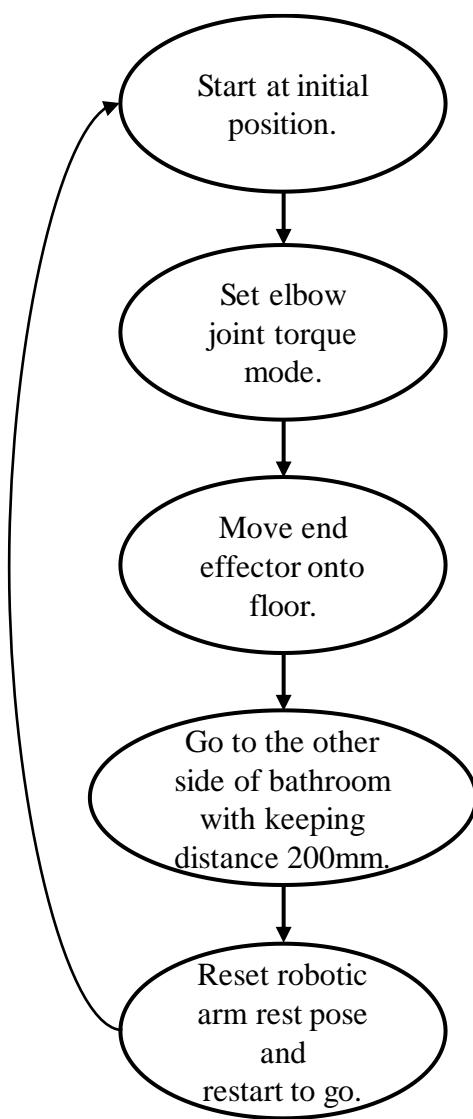


Fig. 4.7 Flow chart for experiment of cleaning floor of bathroom

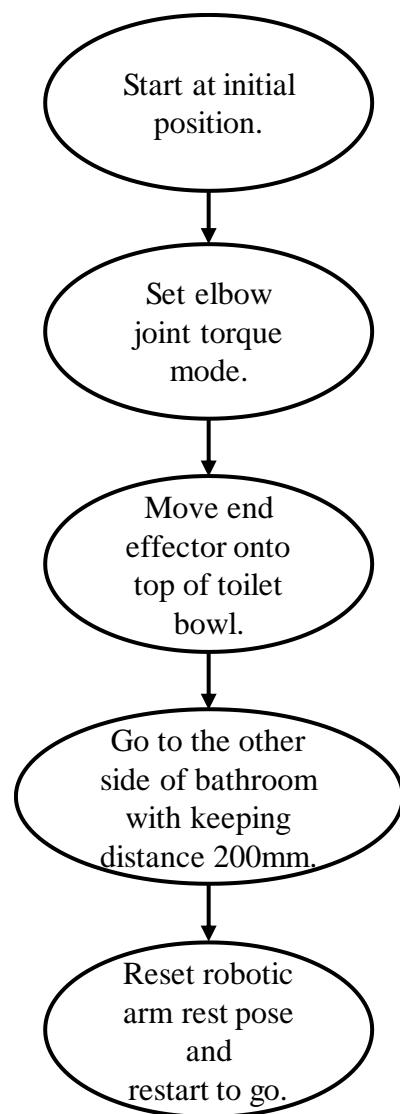


Fig. 4.8 Flow chart for experiment of cleaning top of toilet bowl

4.2 実験結果

ロボットの持つハードウェアの機能として、トイレの汚れを、部屋の 90% の範囲で、統計的に 80% 以上取り除けることが分かった。以下に 3.1 項で示した各実験の結果を述べる。

(1) 便器とロボットの位置関係が取得できるかの検証

ロボットに搭載された 2DLidar は、便器とロボットの位置関係を、ヨー角で -70° から 70° の範囲で取得した。

(2) 5mm 角のトイレットペーパーを取り除けるかの検証

10 回試行した。5mm 角のトイレットペーパー 5 つを取り除いた。

(3) 模擬尿として用意する、水性ペンで描かれた線を取り除けるかの検証

掃除できる面積の内 90% の範囲で 80% 以上汚れを取り除いた。

4.2.1 便器側面の実験

便器側面の清掃具合を示す実験結果を Fig. 4.9 に示す。

簡単な方法、改善された方法をそれぞれ 30 回試行した。水性ペンの汚れが残っていた場合は 1、残っていなかった場合は 0 としてカウントし、30 回の試行の合計が 30 ならその部分は 100% 汚れが残り、0 なら 0% 汚れが残ったとした。

簡単な方法では床から、床からの高さ 100mm の範囲で 20% 以上の拭き残しが出るようになり、最大 100% に達した。改善された方法では床から 100mm の高さまでを検討し、トイレの床から 20mm の高さでは拭き残しが出たが、その部分を除いた、94% の範囲で 86% 汚れを取り除いた。

4.2.2 トイレ床面の実験

床の掃除具合を示す実験結果を Fig. 4.10 に示す。

30 回試行した。水性ペンの汚れが残っていた場合は 1、残っていなかった場合は 0 としてカウントし、30 回の試行の合計が 30 ならその部分は汚れが 100% 残り、0 なら汚れが 0% 残ったとした。

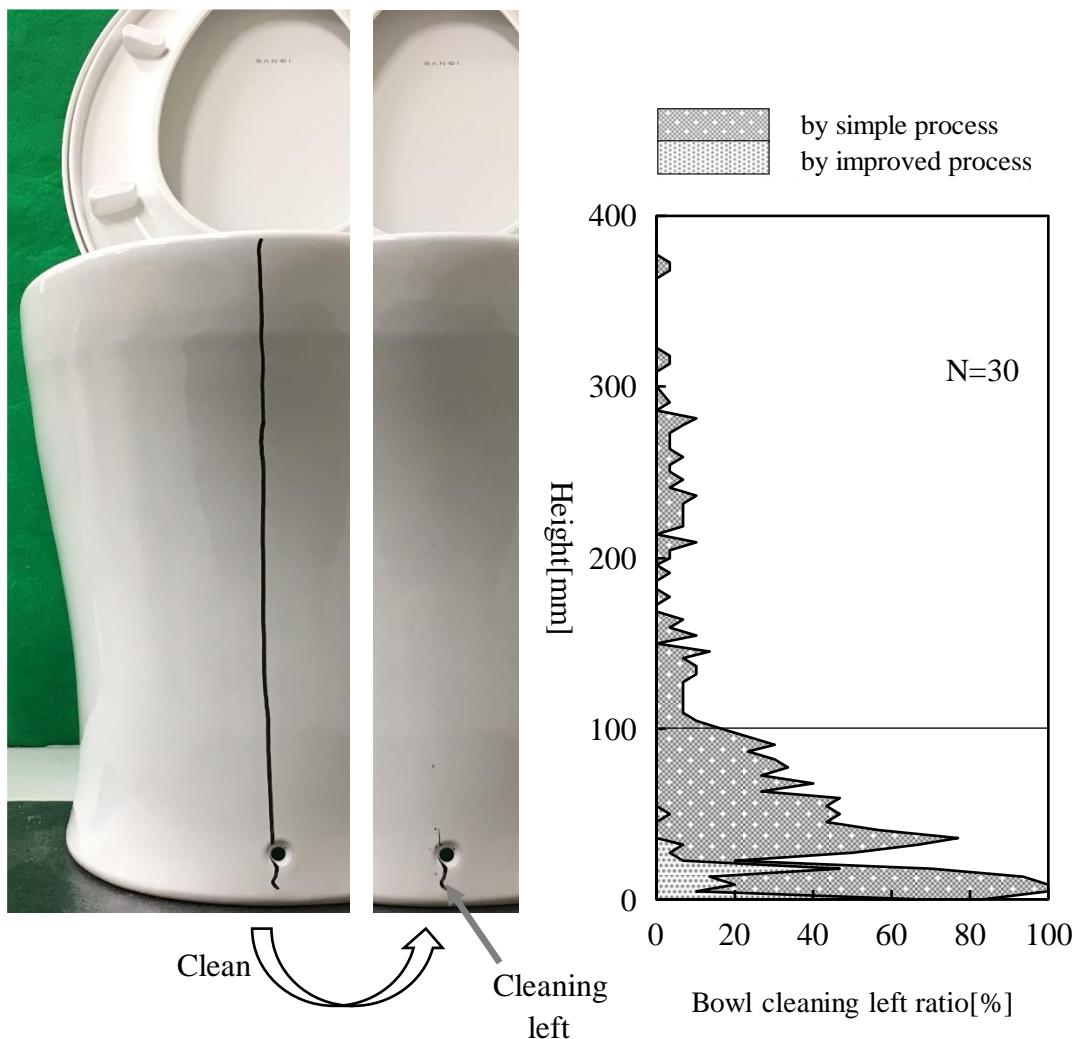


Fig. 4.9 Experiment result of cleaning front of toilet bowl

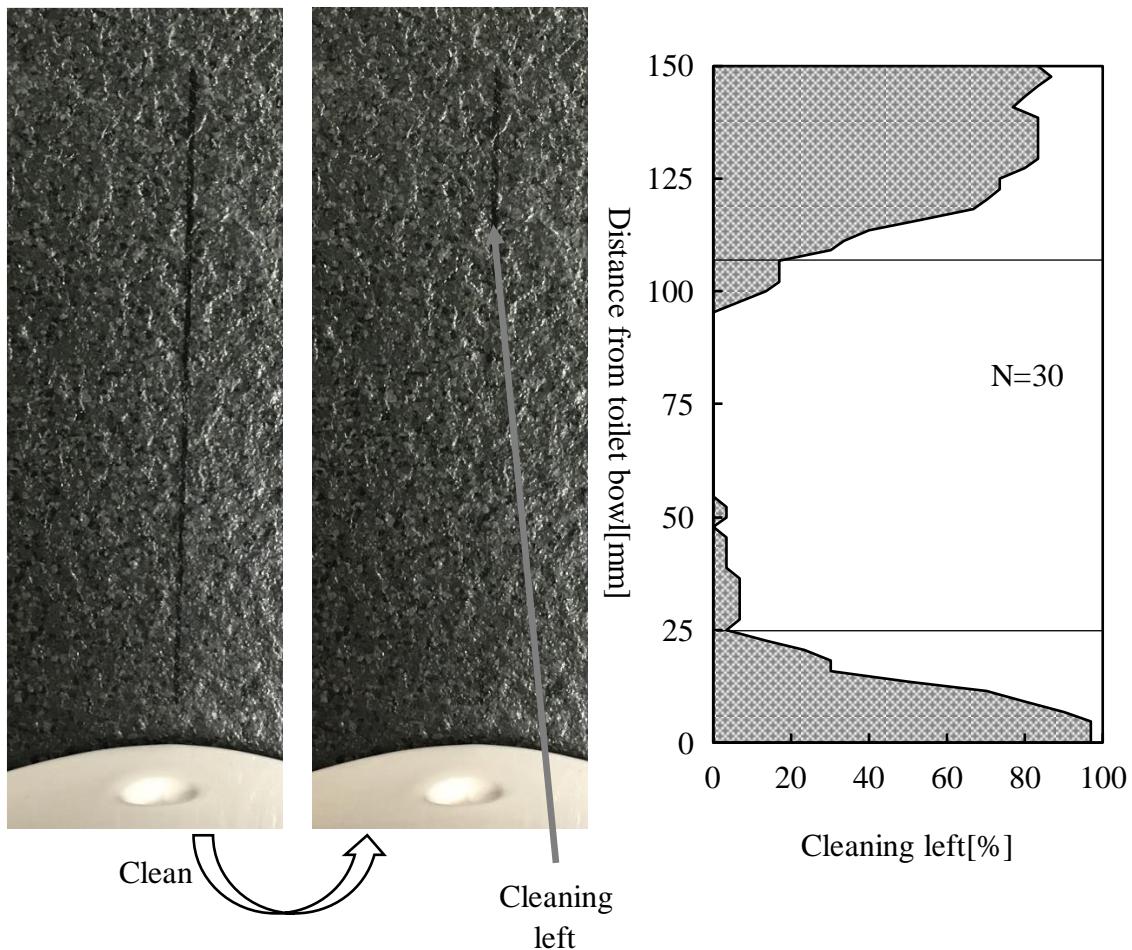


Fig. 4.10 Experiment result of cleaning floor

便器から 25mm までの間と便器から 106mm 以上離れたところで汚れが残った。便器から 25mm 離れた部分から 81mm の範囲で汚れを取り除いた。便器と床の隅から 25mm までの範囲の掃除残しはエンドエフェクタが入り込まなかつたためのものである。便器と床の隅から 106mm 以上離れたところの掃除残しは掃除装置が届かなかつたためのものである。ロボットの掃除経路を変えれば便器と床の隅から 106mm 以上離れたところの掃除残しはなくなるので掃除できる範囲の床面積から、便器から 25mm までの範囲を引いた部分が掃除できる部分である。

トイレ床面の 97% の範囲で約 83% 汚れを落とした。

4.2.3 便器上面の実験

床の掃除具合を示す実験結果を Fig. 4.11 に示す。

15回試行した。水性ペンの汚れが残っていた場合は1、残っていなかった場合は0としてカウントし、10回の試行の合計が10ならその部分は汚れが100%残り、0なら汚れが0%残ったとした。

トイレの半分だけを検証した。90%の範囲で80%汚れを落とした。

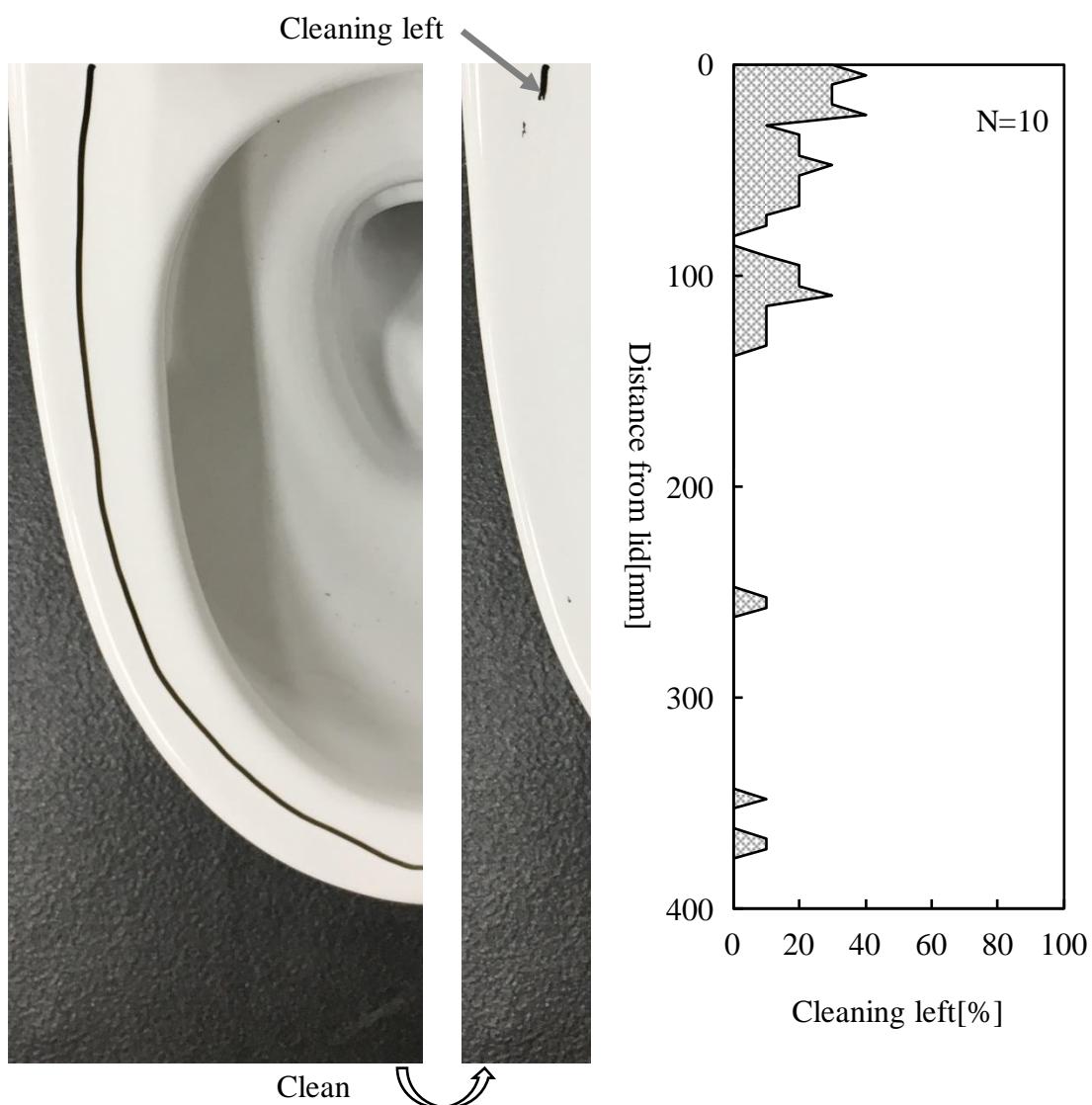


Fig. 4.11 Experiment result of cleaning top of toilet bowl

4.3 考察

便器とロボットの位置関係を取得できるかについて、トイレ内の便器が、壁からどの程度離れているかなどの位置関係が既知であるとすれば、ロボット正面の情報から自己位置推定は十分できる。ロボットが障害物を検出するために必要な距離を Fig. 4.12 に示す。障害物を検出するためには 44mm 離れている必要があり、トイレ部屋はそれより十分大きいので自律ロボットとしても動作できるだろう。

また、トイレの曲面を推定するために搭載したカメラを便器との位置関係の認識に組み合わせることで、狭い空間でもぶつからないよう移動できるだろう。ただし、ここで言う狭い空間は本レポートで用意したトイレモデル程度の大きさで、それより小さい家庭用のトイレ内を移動しまわるにはロボットが大きすぎる。

トイレットペーパーの掃除について、隅に置かれたトイレットペーパーを取り除いた。台車である Roomba の掃除能力を十分発揮できたといえる。ロボットが移動した領域を把握できれば、トイレットペーパーの位置が推定できなくともトイレットペーパーを残さず取り除ける。

便器側面の掃除について、トイレの床から 100mm までの範囲で拭き残しが多かったことには 2 つの理由があると考えた。1 つはトイレ側面にくぼみがあり、くぼみについての汚れを取り除くことができなかったことである。もう 1 つは、トイレの床から 100mm までの範囲ではエンドエフェクタについての掃除道具が便器と床の隅まで届かなかつたことである。

トイレ床面の掃除について、便器から 106mm 以上離れた場所で拭き残しが出たが、それは掃除装置が届かなかつたためである。ロボットが掃除する経路を変えて掃除装置の届く範囲を変えれば掃除できる。例えば、掃除装置の掃除範囲が 81mm であったため、掃除残しをなくすため 40mm ずつ掃除のための経路

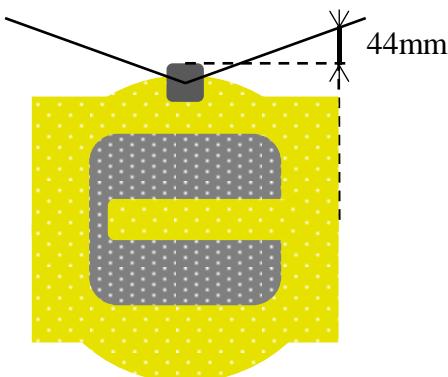


Fig. 4.12 Distance to leave to find obstacles

を移すことが考えられる。

便器上面の掃除について、トイレのフタ側に汚れが多く残った。これは、フタ側まで掃除装置が着いた際に掃除装置をそのままトイレから離したからである。フタ側まで掃除装置が着いたら、ベースジョイントを回転して拭き掃除を行うことで掃除残しはなくなるだろう。また、フタから 200mm 以上離れた位置に残った汚れには Fig. 4.13 に示すような便器内側の汚れがあった。

掃除残しを減らすための工夫は 2 つ考えられる。1 つが便器内側も届くように掃除装置に角度をつける方法である。もう 1 つが便器内側も届くよう掃除装置の弾性を上げる方法である。



Fig. 4.13 Cleaning left inside of toilet bowl

第 5 章 おわりに

トイレ掃除のできるロボットを開発した。乾式清掃に限定してハードウェアの評価を行い、拭き掃除ではトイレの 90% の範囲で、統計的に 80% 汚れを取り除くことができ、ロボットハンドのついた掃除ロボットの有用性を示せた。今後、実用するために残された課題がある。

1 つ目にロボットが大きいことである。家庭用トイレ部屋に対応するためには、直径 150mm に収まることが望ましい。これは家庭用トイレ部屋の大きさが本レポートの実験で用いたトイレ部屋モデルの黒い部分と同じ大きさだと仮定したうえで、ロボットが旋回するときに全方位 50mm の幅を開けるためである。

次に、万が一ロボットが障害物に衝突した際に、ロボットはそれを検出できないことである。本レポートで開発したロボットは自己位置推定に 2DLidar とトイレの表面形状認識に RGB-D Camera を搭載した。ロボット全方位にダンパを使ったタッチセンサをつけることで万が一ロボットが障害物に衝突した際にも直ちに停止できるだろう。

さらに、湿式掃除を達成するために、掃除装置の湿り気を自動的に保つ必要がある。水を撒く方式の湿式掃除は、床が濡れて危ないので避けるべきである。掃除装置は取り外しが簡単だった。掃除装置を洗う必要を考えると、今後も取り外しが簡単なものを採用する必要がある。

今回、ロボットアームの電流値制御で曲面に沿ってエンドエフェクタを動かせたため、RGB-D Camera を使用したシステムの製作などは行われなかった。RGB-D Camera の便器の形状検出は必ずしも必要ではなかった。別の方でロボットアームが動く範囲に障害物があるか検出できれば、ロボットを小型化するうえで、RGB-D Camera をロボットアームにつける必要はない。

電源系統が 3 つに分かれ、管理が難しいことも課題である。知識がない人はできなかった。解決方法としてバッテリーを複数積んで充電ポートを 1 つにする方法やバッテリーを 1 つ積んでバッテリーを外して充電する方法が考えられる。

電源ボタンが外装の中に隠れていて、ロボットの状態を知るためにも外装を外す必要があった。電源ボタンやロボットの状態を示す表示板はロボットの外側に見えるようにする必要がある。

トイレ掃除に限定しないハードウェア構成にしたため、生活支援の環境での応用も期待される。

謝 辞

本プロジェクトにおいて、本学ロボティクス学科出村公成教授には研究の指導だけでなく、RoboCup、World Robot Summitなど多くの挑戦する機会を与えていただきました。心より感謝致します。また、出村研究室の学生にはロボットの仕様や評価法など多くの場面で相談に乗っていただきました。1年間励まし続けていただきました。出村研究室の皆様にお礼を申し上げます。最後にこれまでの学生生活を応援し続けていただいた両親に感謝いたします。

平成 31 年 2 月 13 日

参考文献

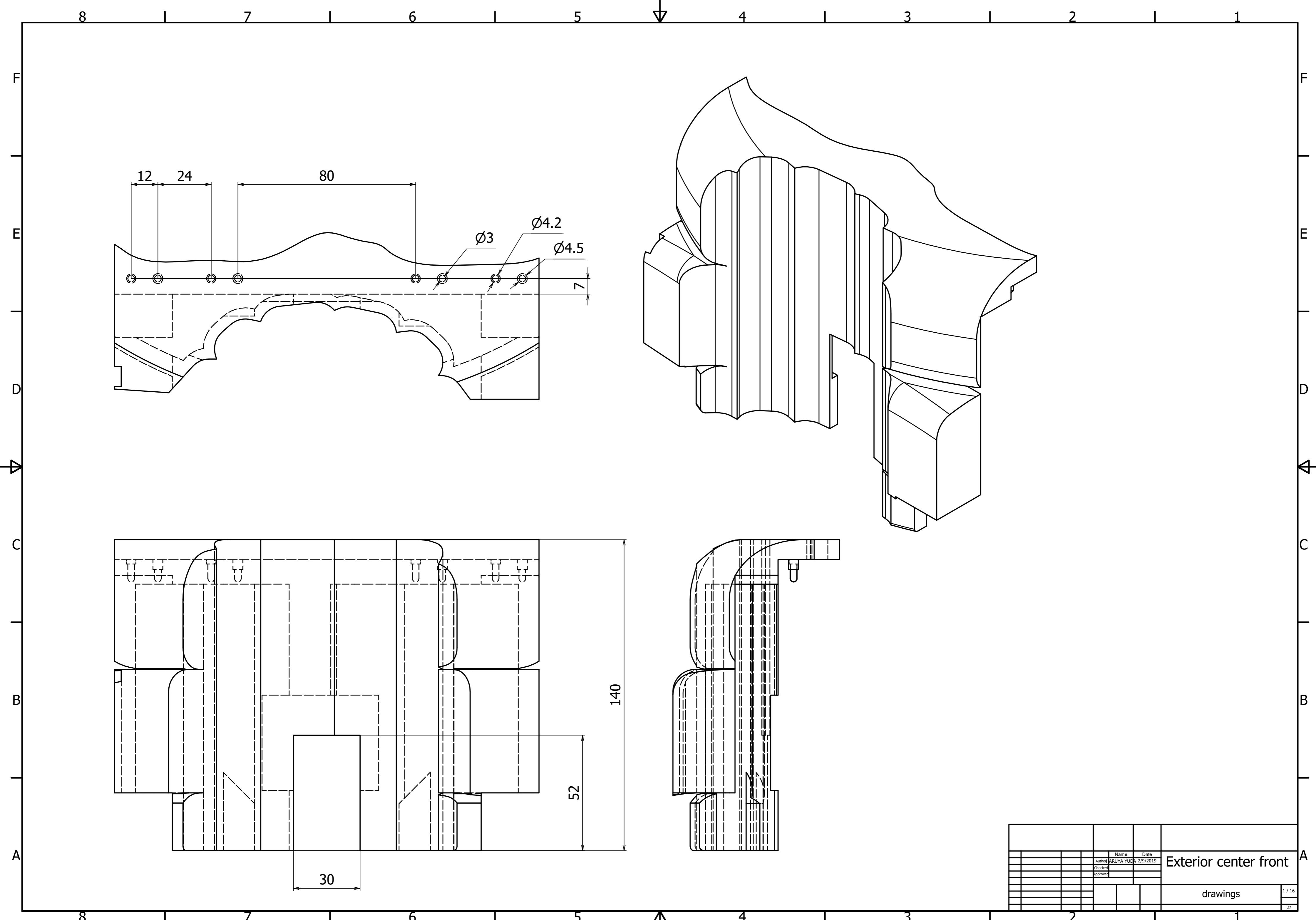
- [1] 宮下, 神田, 塩見, 石田, 萩田：“顧客と顔見知りになるショッピングモール案内ロボット”, 日本ロボット学会誌, vol. 26, no. 7, pp. 821-832, 2008.
- [2] C.G. Burgar, P.S. Lum, P.C. Shor, H.F.M. Van Der Loos: “Development of robots for rehabilitation therapy: The Palo Alto VA/Stanford experience,” Journal of Rehabilitation Research and Development37, pp. 663-673, 2000.
- [3] 石井：“食事支援ロボット「マイスプーン」”, 日本ロボット学会誌, vol.21, no. 4, pp.378-381, 2003.
- [4] 薩見, 青山, 石川, 関, 足立, 石村, 高橋, 横田：“トイレ用小型清掃ロボットの開発”, 日本ロボット学会誌, vol. 29, no. 7, pp. 573-583, 2011.
- [5] 青山, 田島, 横田, 尾崎, 山本：“自立走行式床面掃除ロボットの開発”, 日本ロボット学会誌, vol. 16, no. 1, pp. 57-64, 1998.
- [6] T. Miyake, H. Ishihara, and R. Shoji, “Development of small-size window cleaning robot by wall climbing mechanism,” International Symposium on Automation and Robotics in Construction 2006, Tokyo, Japan, 2006, pp. 215-220.
- [7] <http://www.faculty.ait.ac.th/pisut/PDD/PDD06/PDD06G1.pdf>, 参照日: 2019年1月26日.
- [8] <https://www.kickstarter.com/projects/1209829383/spinx-worlds-first-toilet-cleaning-robot>, 参照日: 2019年1月26日.
- [9] <https://www.irobot.com/for-the-home/vacuuming/roomba>, 参照日: 2019年1月26日.

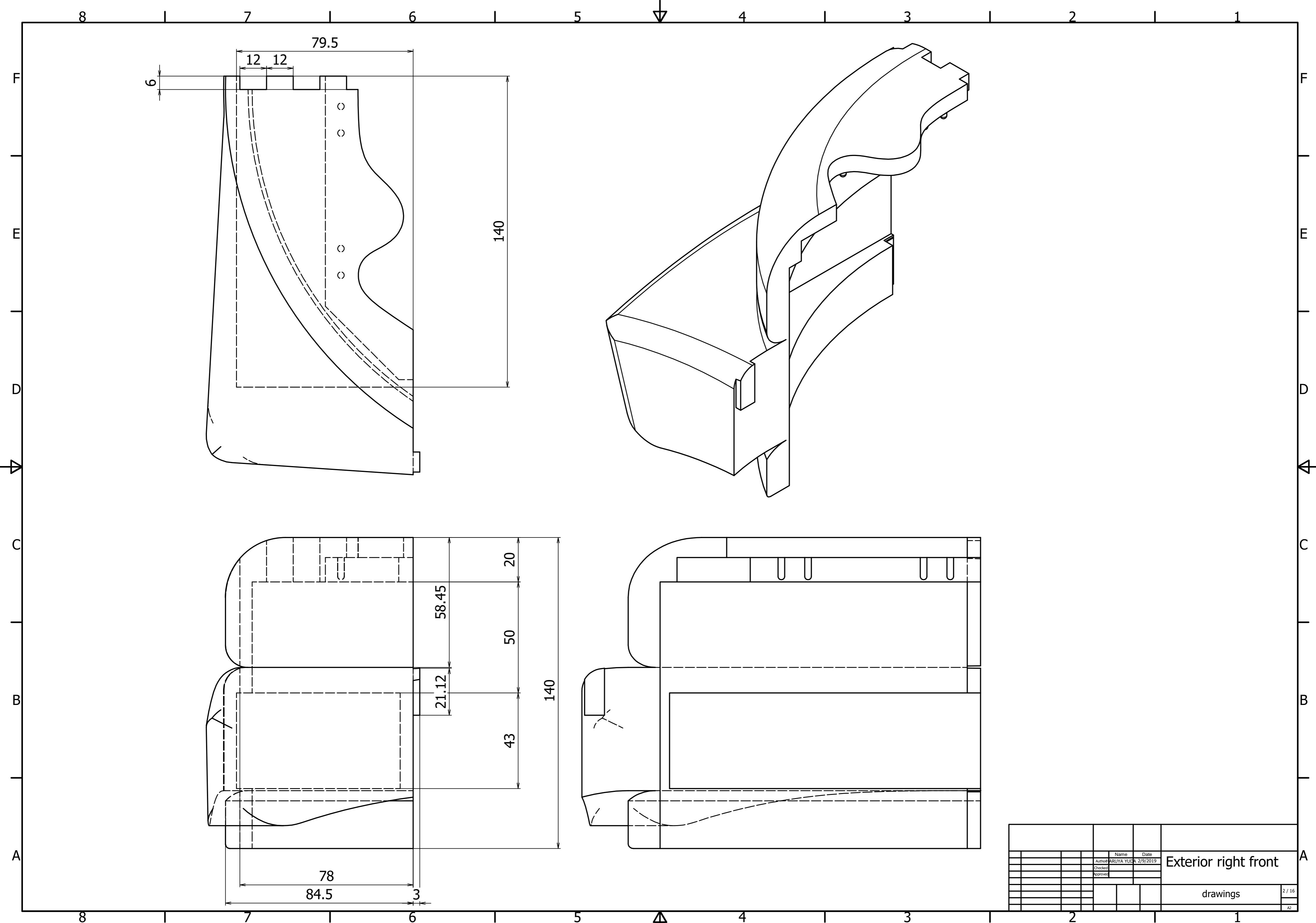
付録

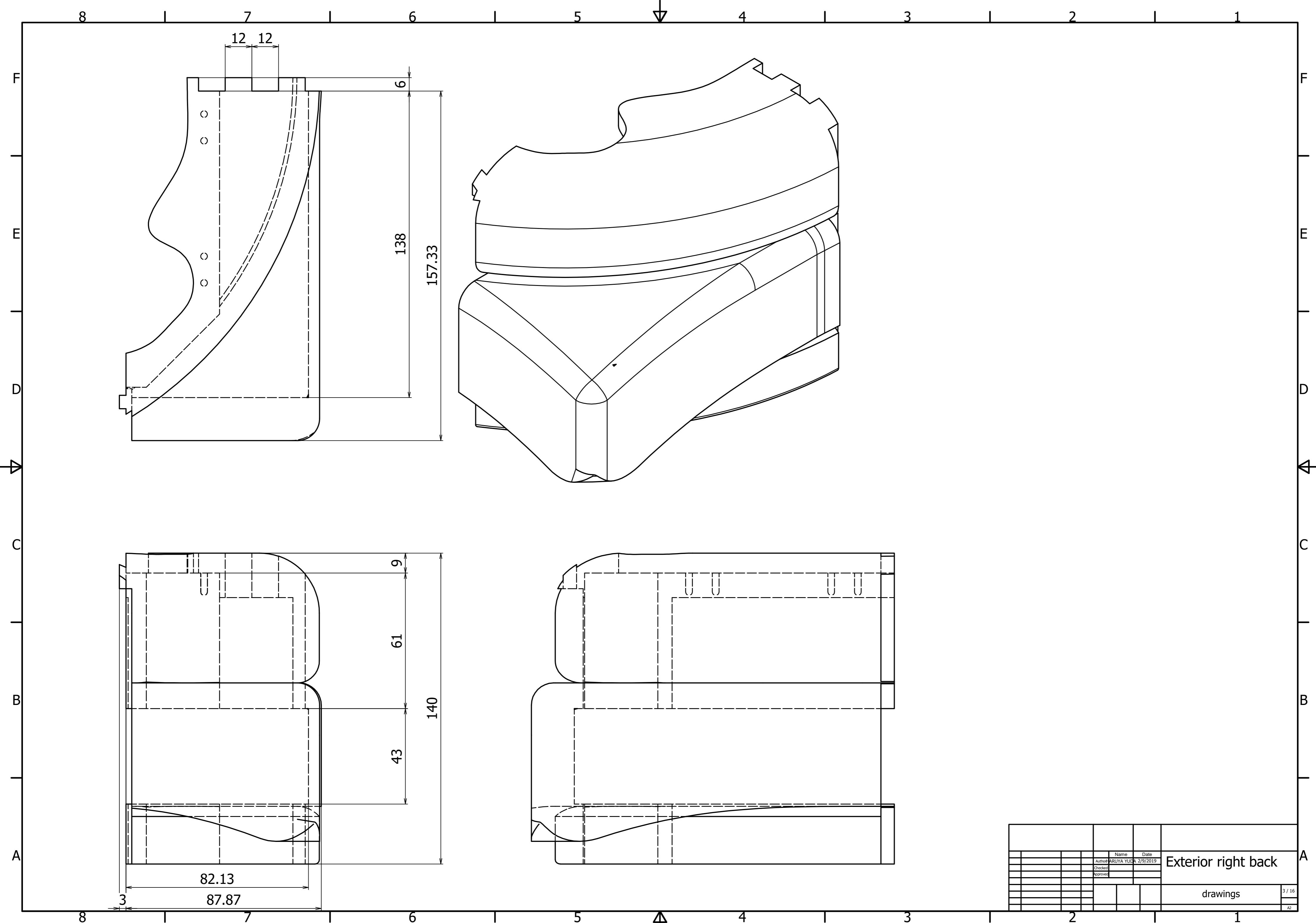
本プロジェクトで作成したソースコードと設計図面を示す。

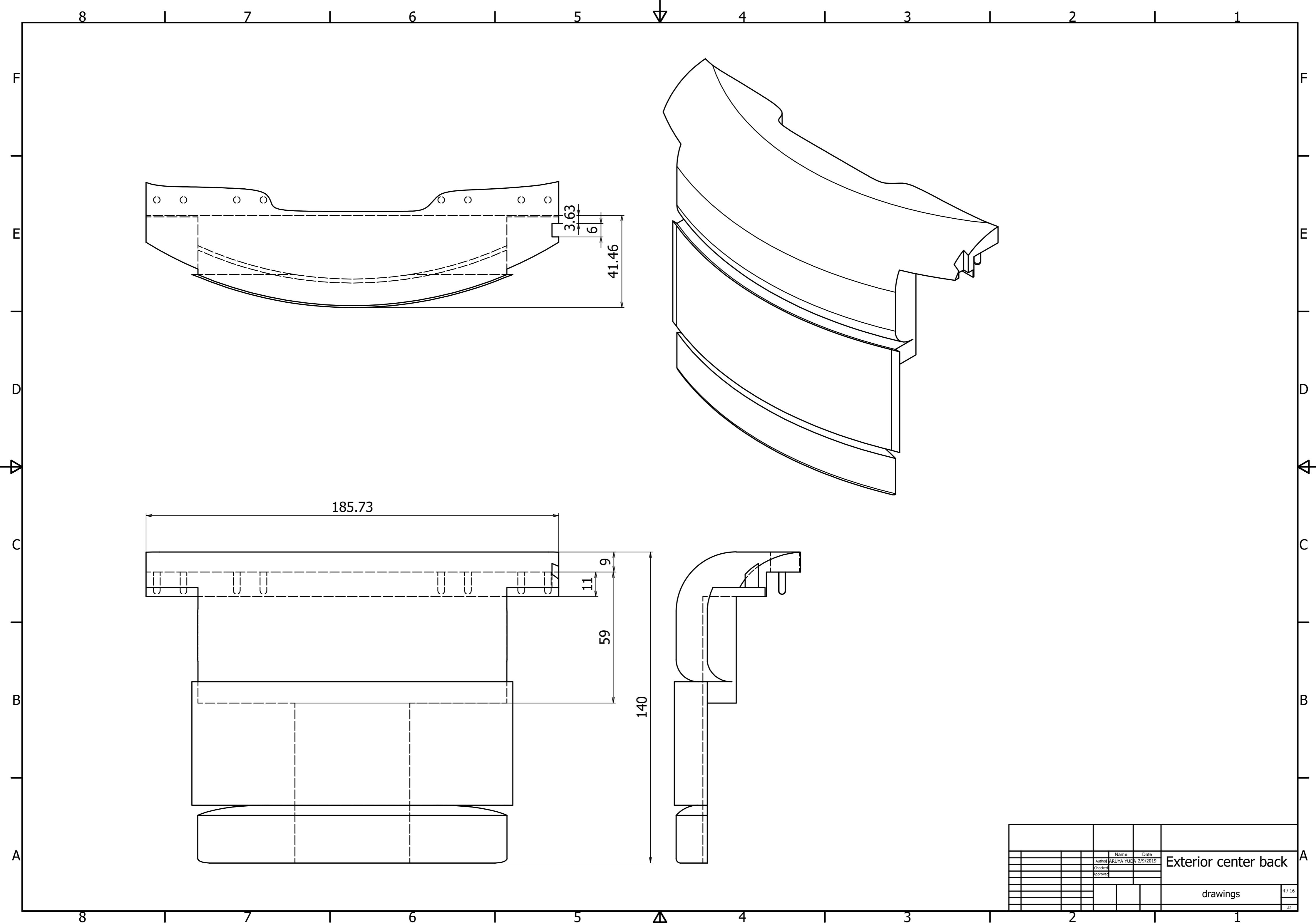
- パーツリスト
- 設計図面
- dead_recognition.cpp

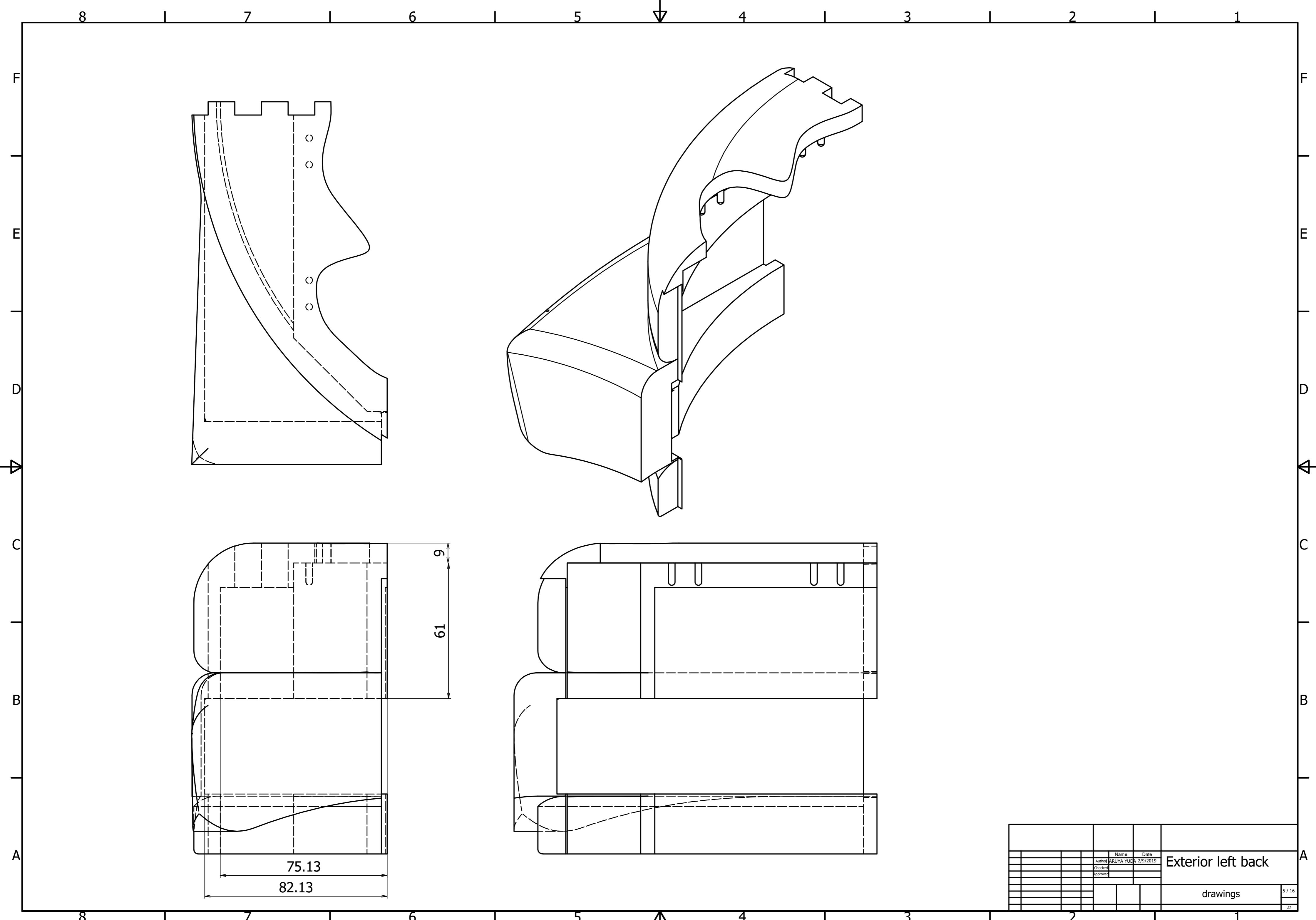
Parts	Number
Exterior center front	1
Exterior right front	1
Exterior right back	1
Exterior center back	1
Exterior left back	1
Exterior left front	1
Lidar base	1
Robotic arm base	1
RGB-D camera base	1
Emergency switch base	1
Turtlebot plate	1
Structure 1	2
Structure 2	2
Structure 3	2
Structure 4	4
Structure connector	14
Mikata Arm	1
HOKUYO 2DLidar UTM30LX	1
INTEL REALSENSE DEPTH CAMERA D435	1

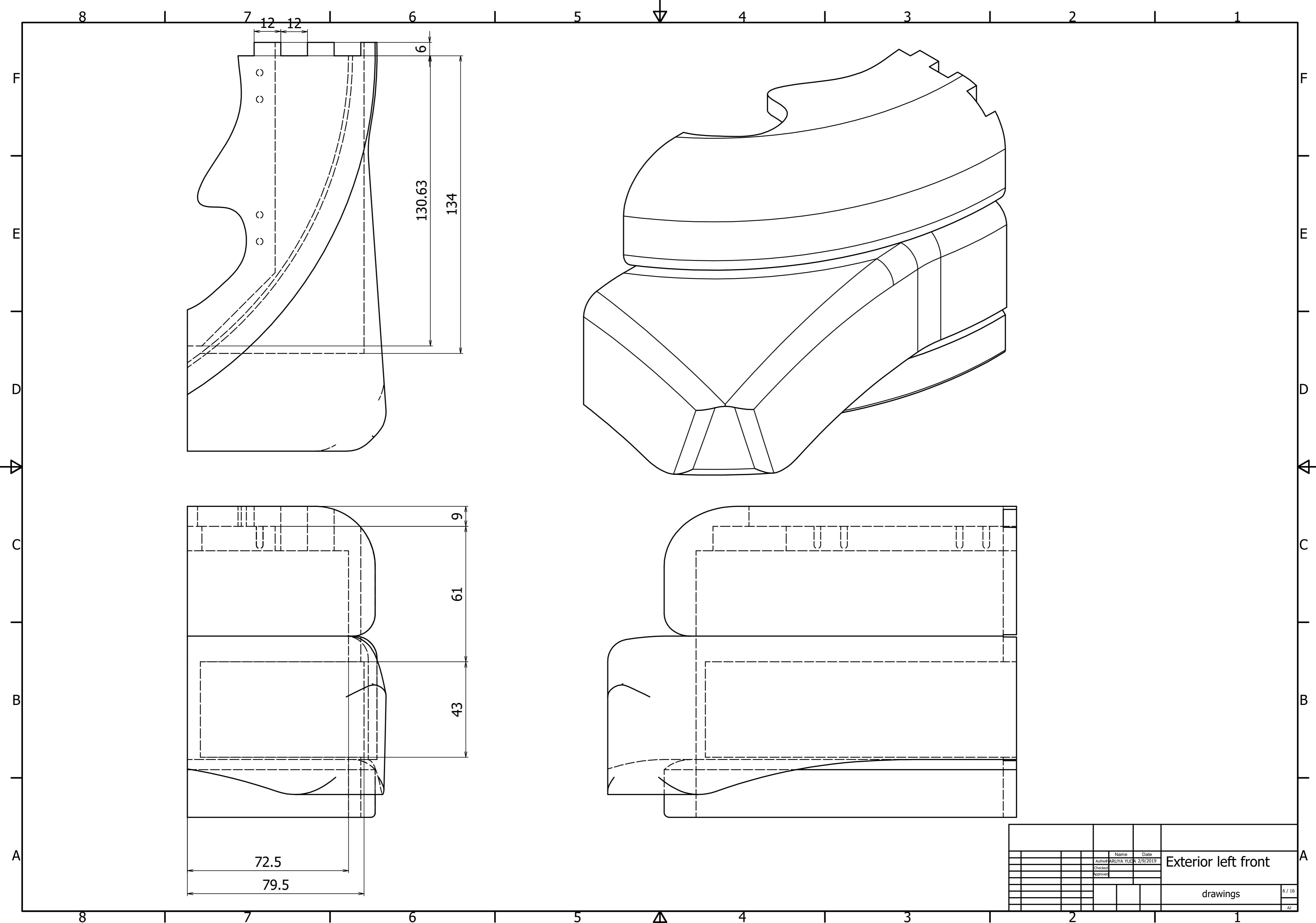


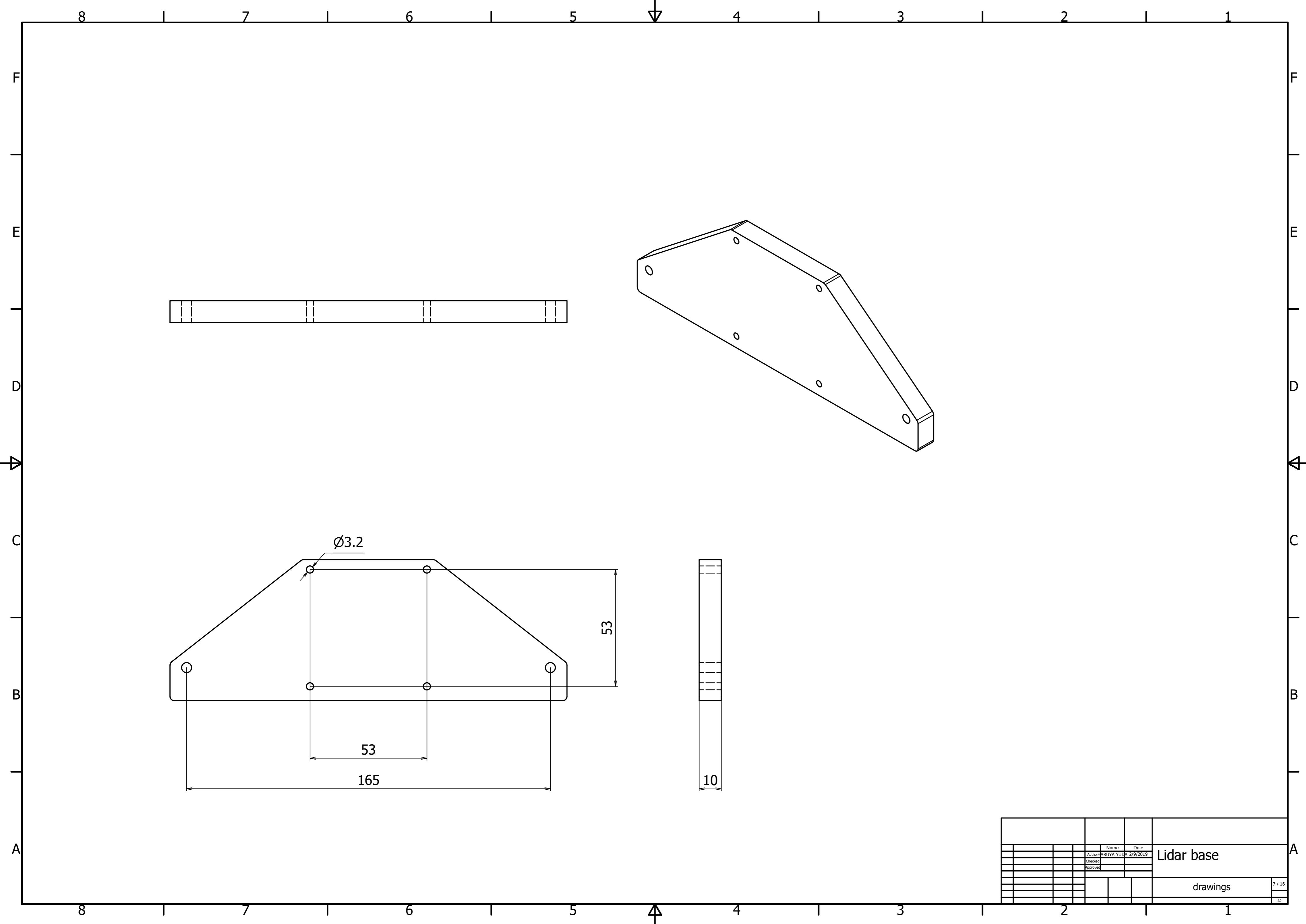


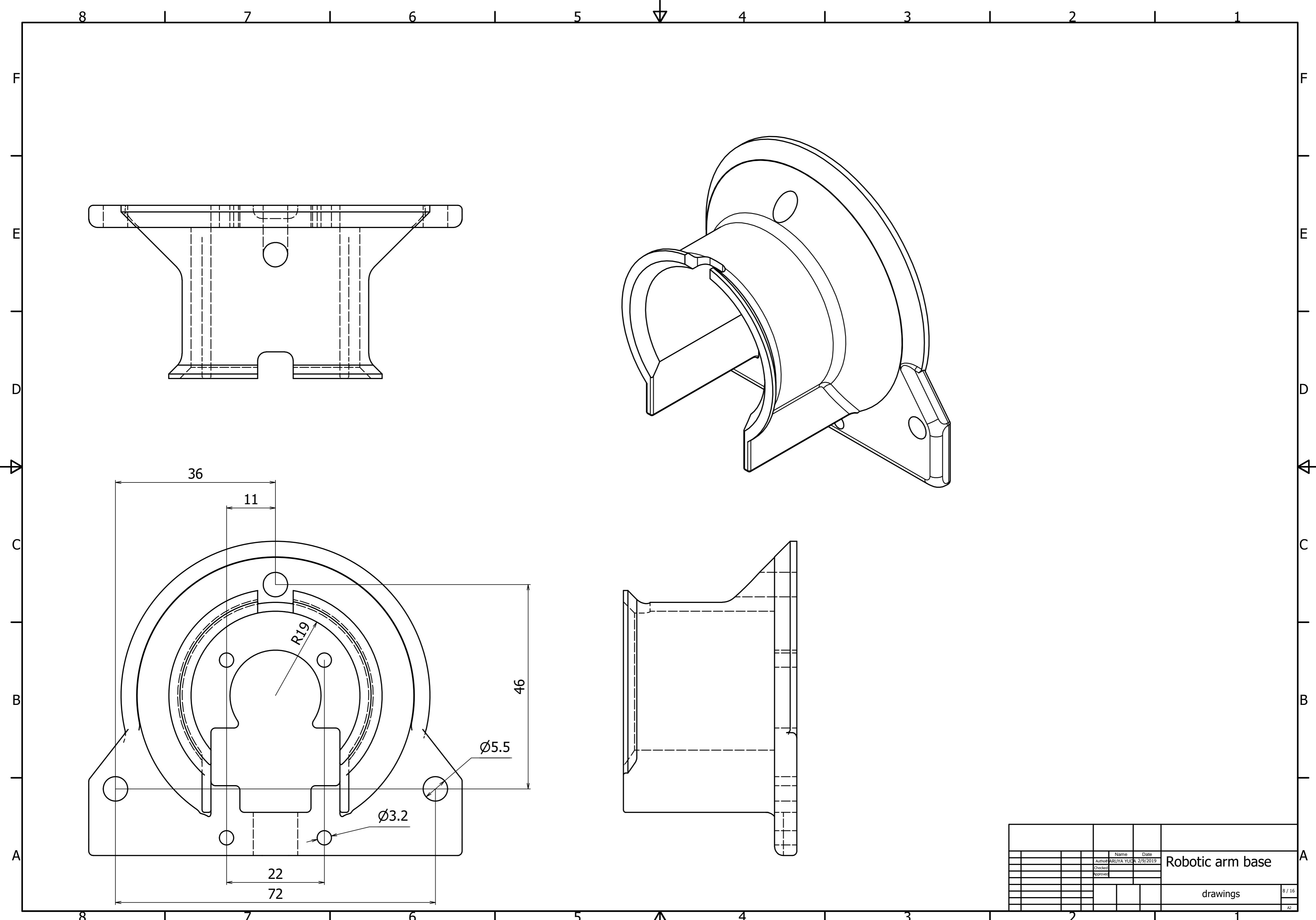


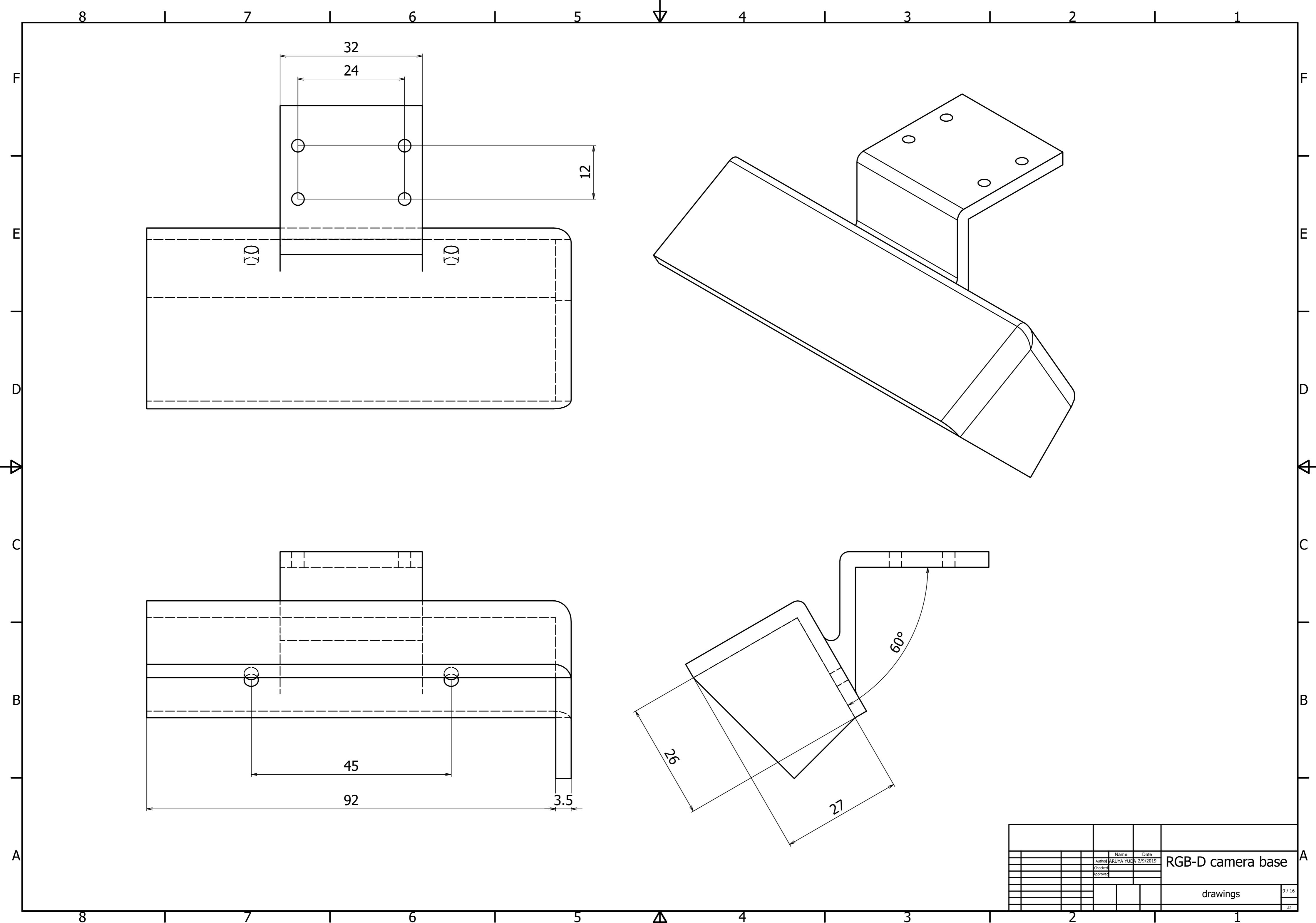


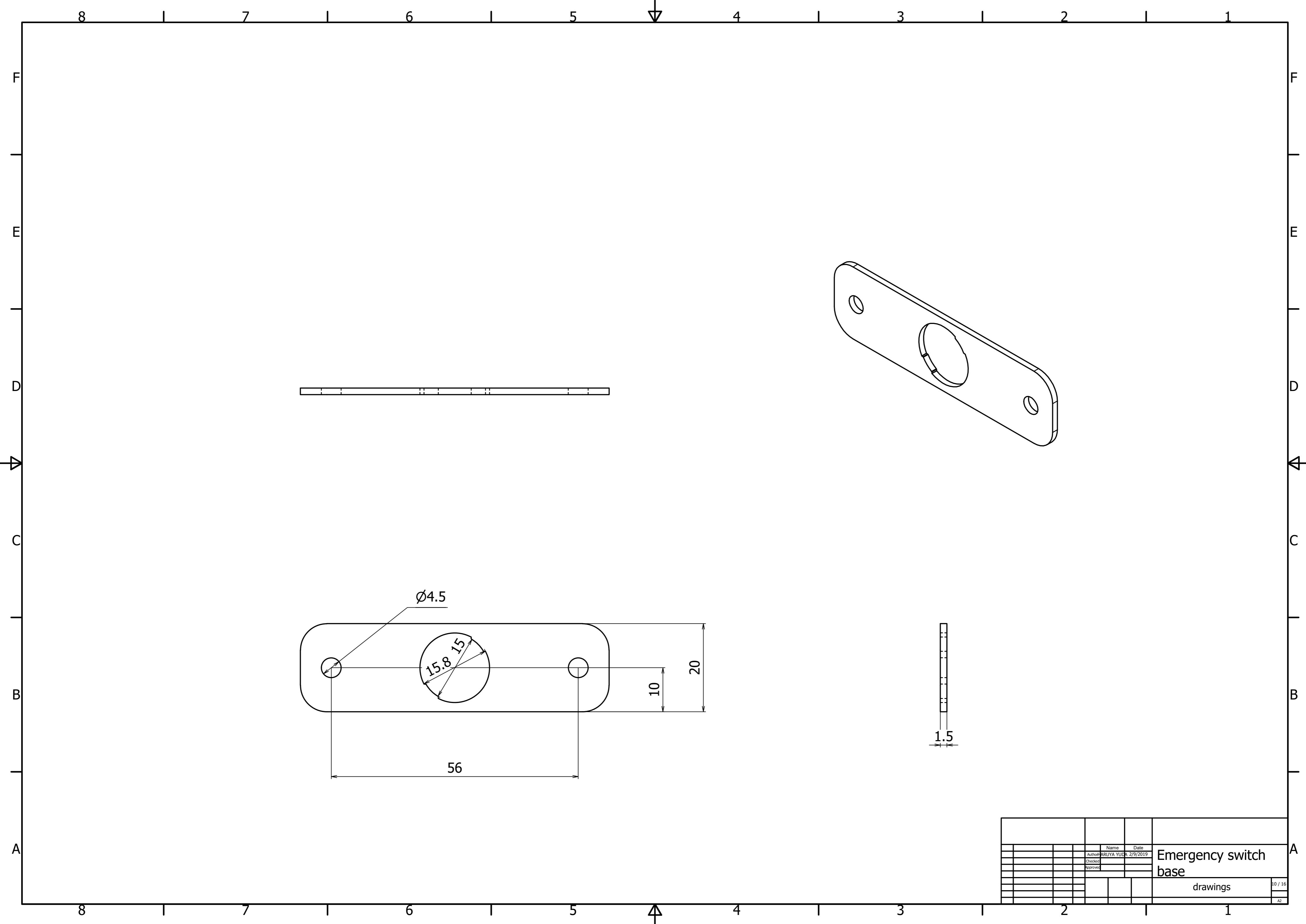


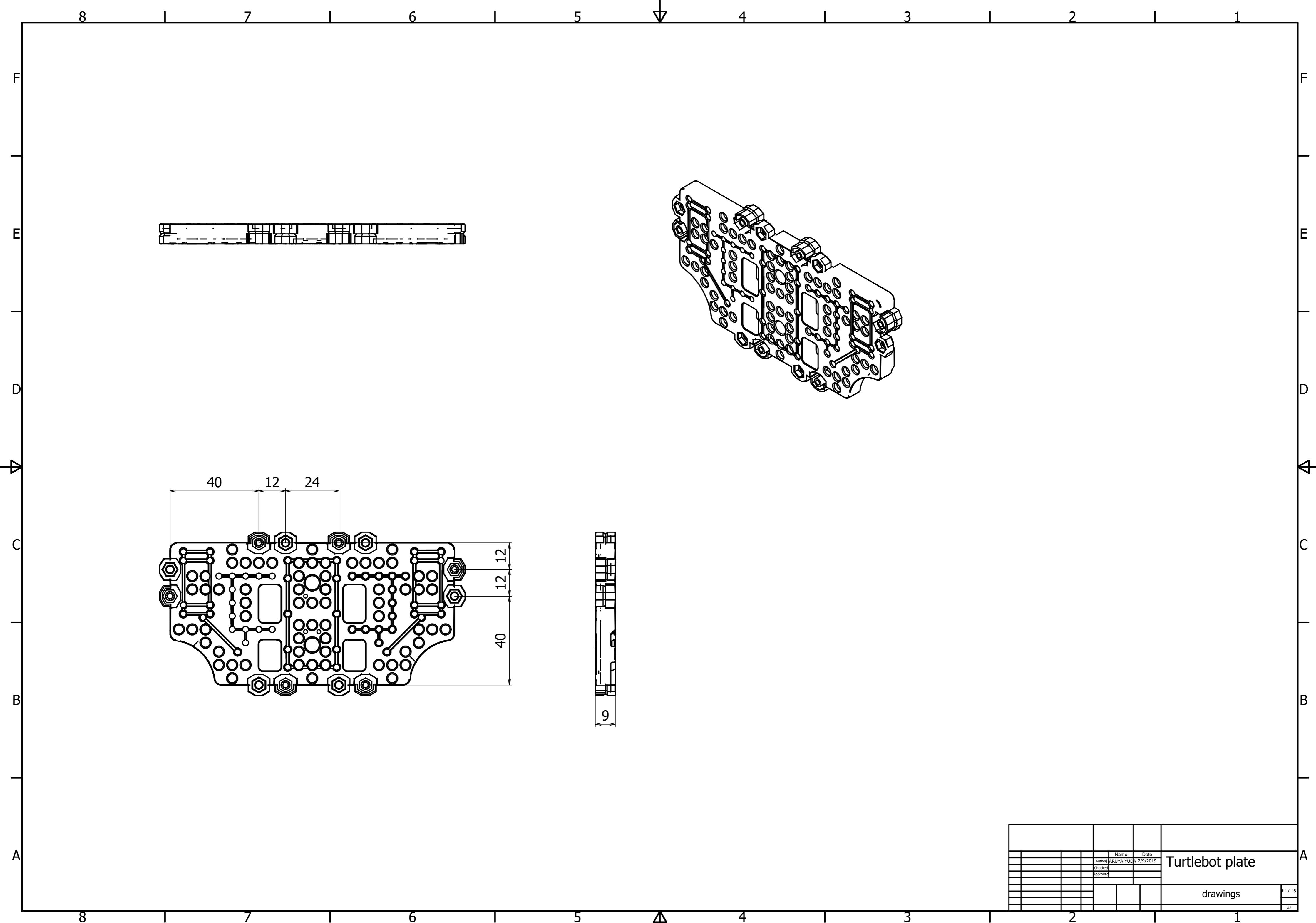


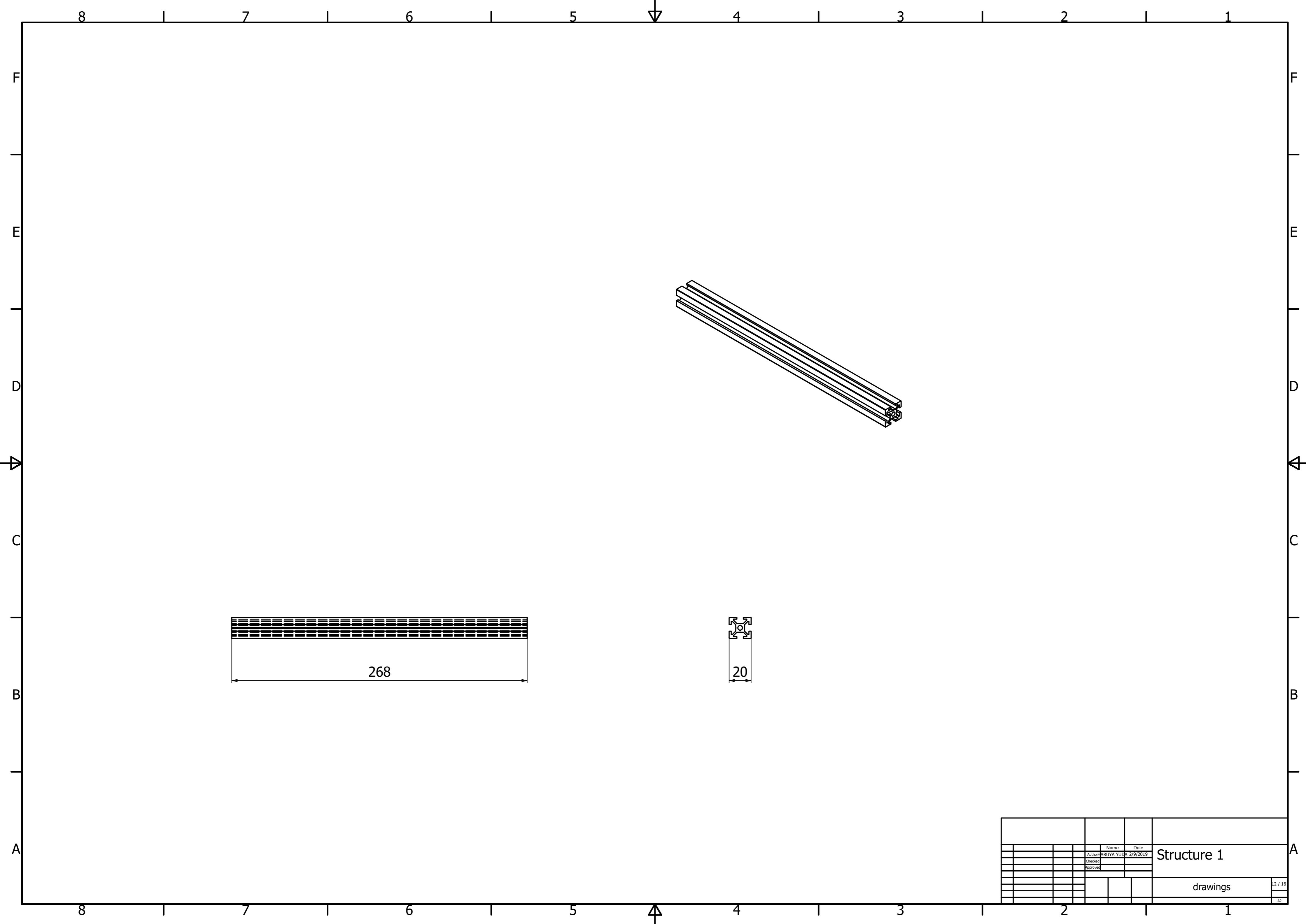


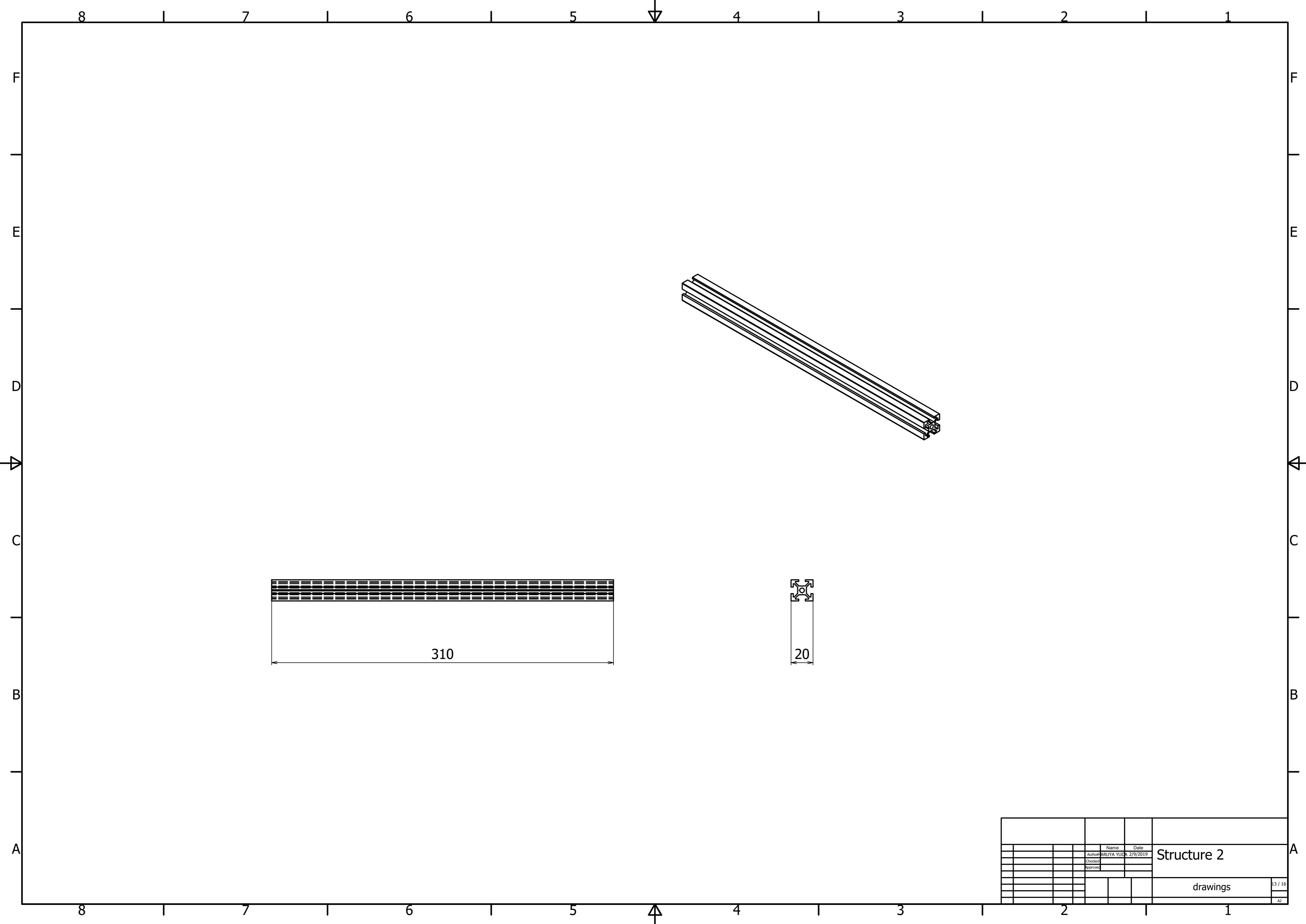












8

1

1

1

1

1

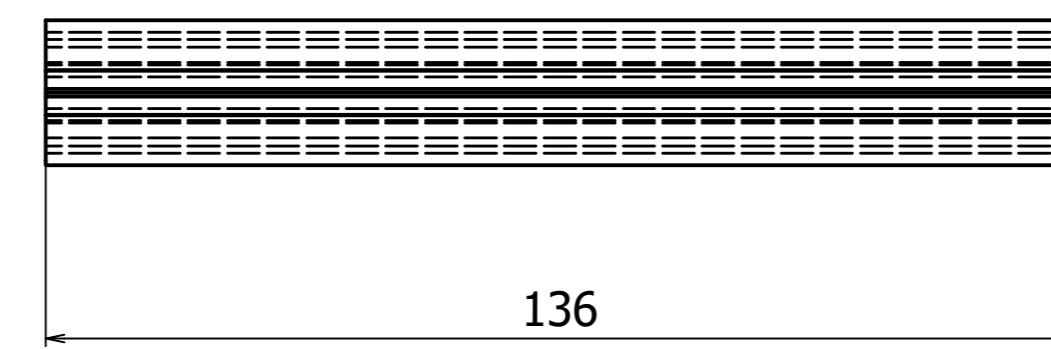
1

1

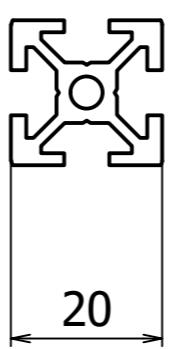
1

1

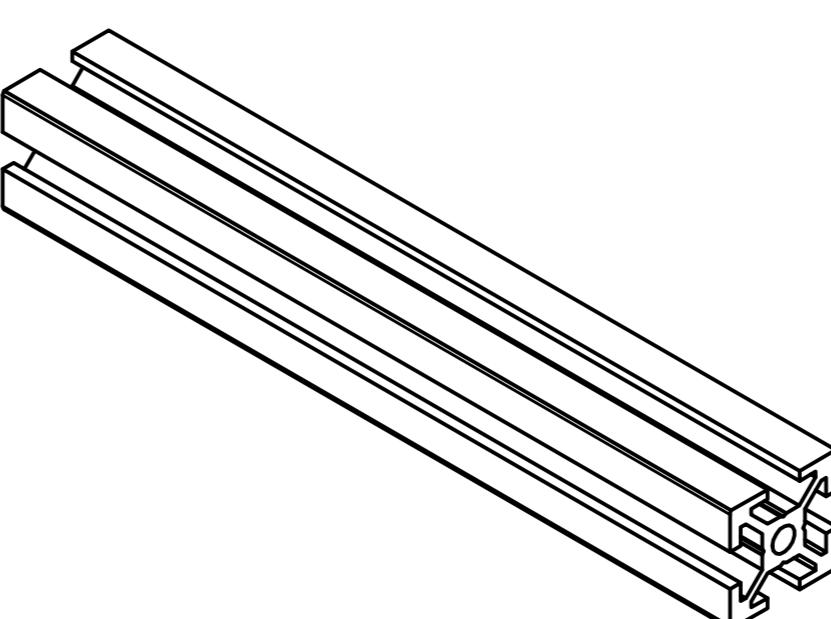
A B C D E F

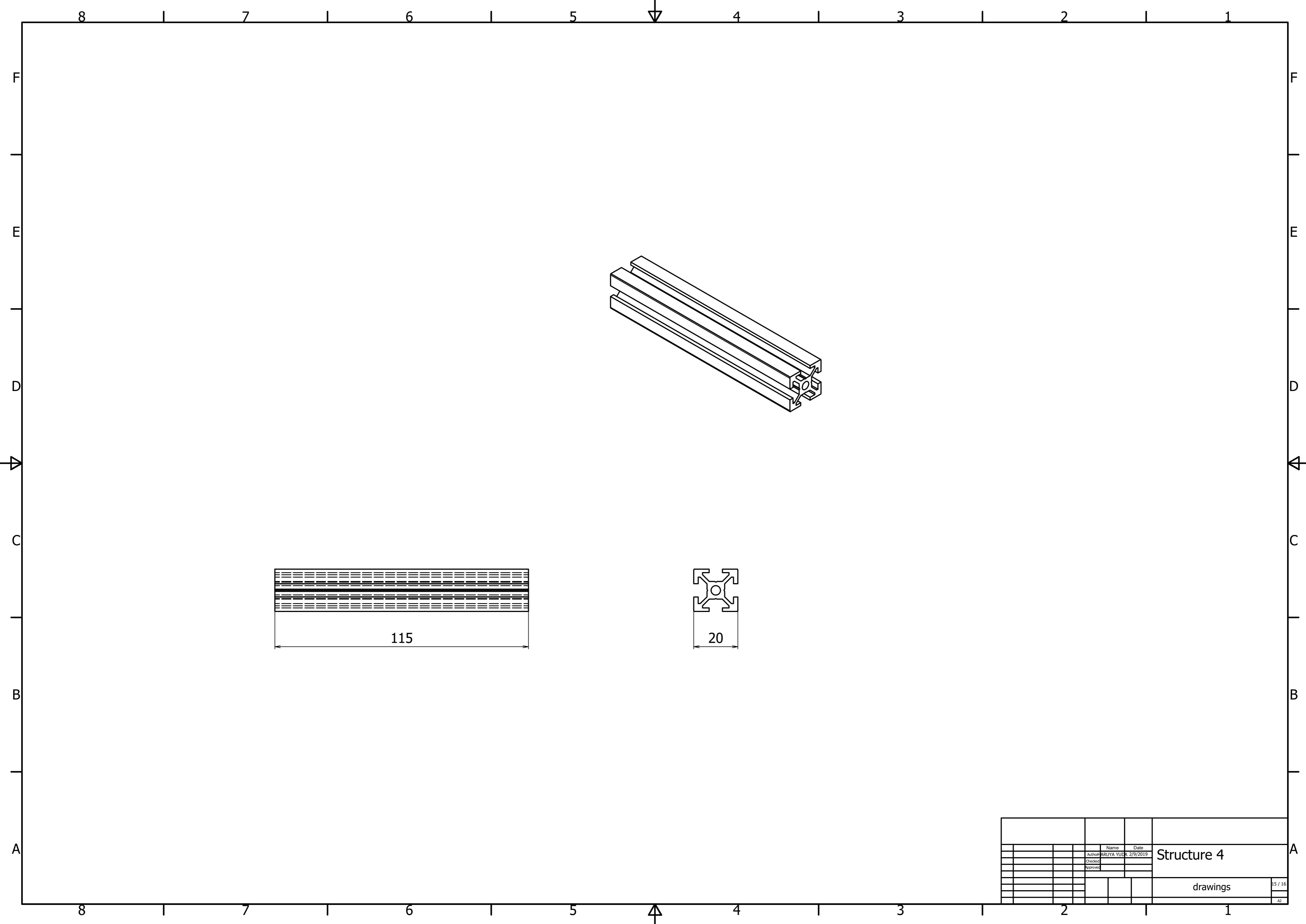


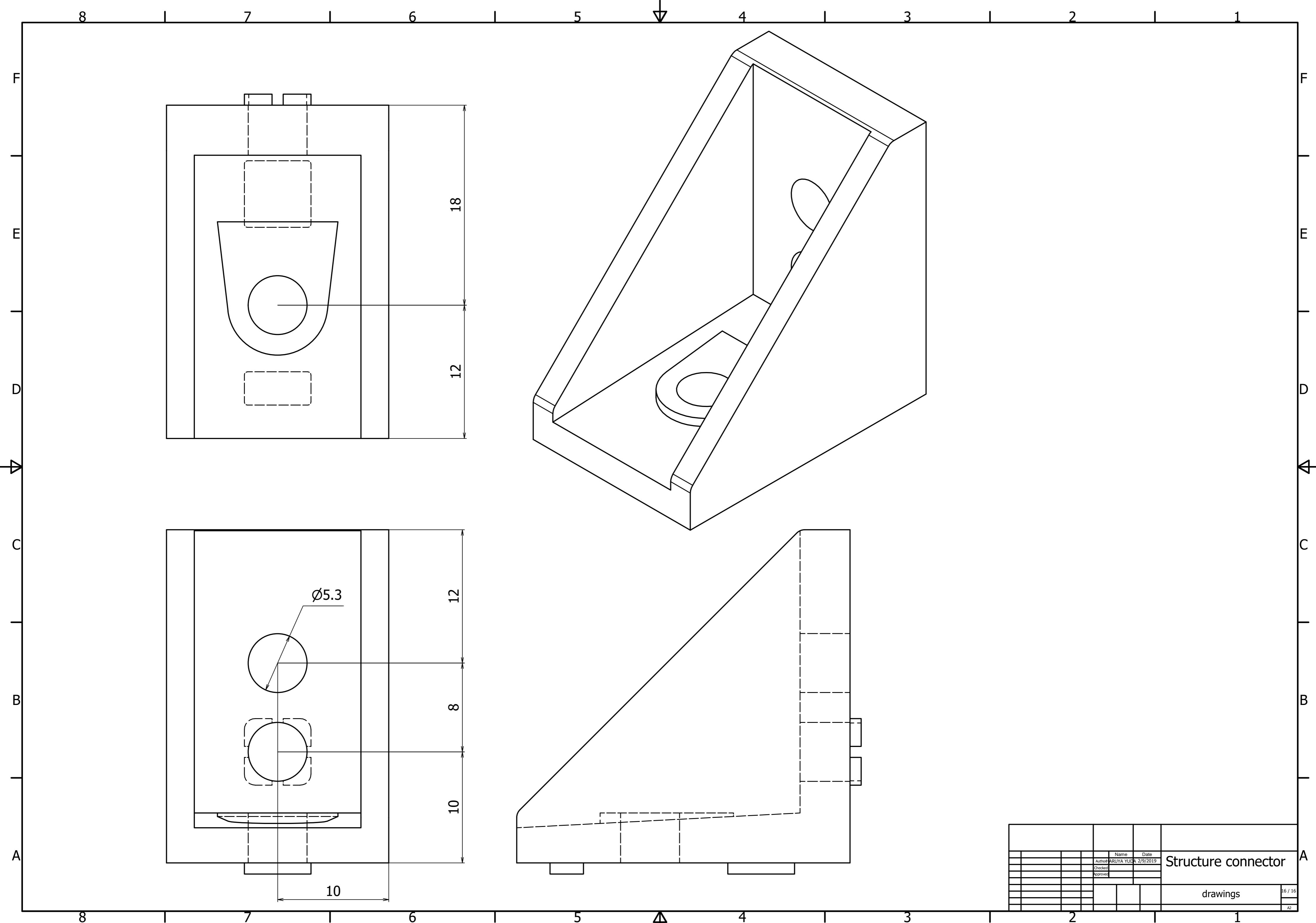
136



20







dead_recognition.cpp

```
1 //*****
2 /*This code was used at WRS. */  
3 /*You can clone it from DemuLab's private repository*/  
4 /*"https://github.com/demulab/happy_burger3.git" */  
5 //*****  
6 #include <ros/ros.h>  
7 #include <tf/tf.h>  
8 #include <geometry_msgs/Twist.h>  
9 #include <geometry_msgs/Pose2D.h>  
10 #include <nav_msgs/Odometry.h>  
11 #include <math.h>  
12 #include <std_msgs/Bool.h>  
13 #include <iostream>  
14 #include <sstream>  
15 #include "arm.hpp"  
16 #include <vector>  
17  
18 using namespace std;  
19  
20 geometry_msgs::Pose2D current_pose;  
21 ros::Publisher pose2d_pub;  
22 ros::Publisher twist_pub;  
23 geometry_msgs::Twist twist;  
24  
25 struct MyPose {  
26     double x;  
27     double y;  
28     double yaw;  
29 };  
30  
31 MyPose waypoint[] = {  
32     { 0.40,  0.00,  0.0 * M_PI/180},//0  
33     { 1.05,  0.00, -90 * M_PI/180},//1  
34     { 1.05, -0.80, -90 * M_PI/180},//2  
35     { 1.05, -0.80, -180 * M_PI/180},//3  
36     { 0.40, -0.80, -180 * M_PI/180},//4  
37     { 0.00, -0.80,  0.0 * M_PI/180},//5  
38     { 0.40, -0.80,  0.0 * M_PI/180},//6  
39     { 1.05, -0.80,   90 * M_PI/180},//7  
40     { 1.05,   0.0,  180 * M_PI/180},//8  
41     { 0.90,   0.0,  180 * M_PI/180},//9  
42     { -0.1,   0.1,  0.0 * M_PI/180},//10  
43     { qqq,     qqq,           qqq}};  
44  
45 // cleaning right  { 0.36809,  0.3619, -1.31161,  1.77482, -0.57369}  
46 // super man       {-0.04136,-0.9142,-0.04451,-0.66724,-0.57369}  
47 // cleaning left   { 2.90684,  0.4632, -1.09066,  1.57334, -0.57369}  
48 std::vector<std::vector<double>> armpose ={  
49     { 0.36809,  0.3619, -1.31161,  1.77482, -0.57369},//0
```

```

50 { 0.36809, 0.3619,-1.31161, 1.77482,-0.57369},//1
51 { 0.36809, 0.3619,-1.31161, 1.77482,-0.57369},//2
52 { 0.36809, 0.3619,-1.31161, 1.77482,-0.57369},//3
53 {-0.04136,-0.9142,-0.04451,-0.66724,-0.57369},//4
54 {-0.04136,-0.9142,-0.04451,-0.66724,-0.57369},//5
55 { 2.90684, 0.4632,-1.09066, 1.57334,-0.57369},//6
56 { 2.90684, 0.4632,-1.09066, 1.57334,-0.57369},//7
57 { 2.90684, 0.4632,-1.09066, 1.57334,-0.57369},//8
58 {-0.04136,-0.9142,-0.04451,-0.66724,-0.57369},//9
59 {-0.04136,-0.9142,-0.04451,-0.66724,-0.57369},//10
60 {-0.04136,-0.9142,-0.04451,-0.66724,-0.57369},//11
61 };
62
63 const double rot_vel = 0.3; // angular velocity [rad/s]
64 const double linear_vel = 0.2; // linear velocity [m/s]
65 const double kp_linear = 0.75; // proportional gain for linear velocity
66 const double kp_angular = 1.0; // proportional gain for linear velocity
67 const double ki_angular = 0.05; // proportional gain for linear velocity
68 const double rot_vel_i = 0.1;
69
70 void odomCallback(const nav_msgs::OdometryConstPtr& msg)
71 {
72     // linear position
73     current_pose.x = msg->pose.pose.position.x;
74     current_pose.y = msg->pose.pose.position.y;
75
76     // quaternion to RPY conversion
77     tf::Quaternion q(
78         msg->pose.pose.orientation.x,
79         msg->pose.pose.orientation.y,
80         msg->pose.pose.orientation.z,
81         msg->pose.pose.orientation.w);
82     tf::Matrix3x3 m(q);
83     double roll, pitch, yaw;
84     m.getRPY(roll, pitch, yaw);
85
86     // angular position
87     current_pose.theta = yaw;
88     pose2d_pub.publish(current_pose);
89 }
90
91 // Set the direction
92 void gotoDirection(int no)
93 {
94     double thresh = 1.0 * M_PI/180.0;
95     double diff = thresh + 1; // make diff is bigger than thresh
96
97     ros::Rate loop_rate(100);
98     double angle_tmp_i = 0;
99     twist.linear.x = 0; // linear velocity is 0, i.e., only rotate

```

```

100    while (fabs(diff) > thresh) {
101        diff = waypoint[no].yaw - current_pose.theta;
102        if (diff > M_PI) diff -= 2 * M_PI;
103        else if (diff < -M_PI) diff += 2 * M_PI;
104        // cout << "dir diff:" << diff << endl;
105        // p
106        double angle_tmp_p = kp_angular * diff;
107        // i
108        angle_tmp_i += ki_angular*diff;
109        if (angle_tmp_i >= rot_vel_i ) angle_tmp_i=rot_vel_i;
110        else if (angle_tmp_i <= -rot_vel_i) angle_tmp_i=-rot_vel_i;
111
112        double angle_tmp = angle_tmp_p + angle_tmp_i;
113        if (angle_tmp >= rot_vel ) angle_tmp=rot_vel;
114        else if (angle_tmp <= -rot_vel) angle_tmp=-rot_vel;
115
116        twist.angular.z = angle_tmp;
117
118        twist_pub.publish(twist);
119        ros::spinOnce();
120        loop_rate.sleep();
121    }
122    twist.linear.x = linear_vel;
123    twist.angular.z = 0;
124    twist_pub.publish(twist);
125    ros::spinOnce();
126}
127
128 // Go to the next waypoint
129 void gotoPosition(int no)
130 {
131     double thresh = 0.01; // [m]
132     // initial diff_dist should be bigger than thresh
133     double diff_dist = thresh + 1;
134     twist.linear.x = linear_vel;
135
136     ros::Rate loop_rate(100);
137
138     double angle_tmp_i = 0;
139     while (fabs(diff_dist) > thresh) {
140         double diff_x = waypoint[no].x - current_pose.x;
141         double diff_y = waypoint[no].y - current_pose.y;
142         double diff_theta = atan2(diff_y, diff_x) - current_pose.theta;
143         diff_dist = sqrt(diff_x * diff_x + diff_y * diff_y);
144         cout << "gotoPosition" << diff_dist << endl;
145         cout << "diff dist:" << diff_dist << "[m]" << endl;
146         cout << "current_pose.x:" << current_pose.x << "[m]" << endl;
147         cout << "current_pose.y:" << current_pose.y << "[m]" << endl;
148         cout << "current_pose.theta:" << current_pose.theta << "[rad]" << endl;
149         if (diff_theta > M_PI) diff_theta -= 2 * M_PI;

```

```

150     else if (diff_theta < - M_PI) diff_theta += 2 * M_PI;
151     //maximum velosity
152     double angle_tmp_p = kp_angular * diff_theta;
153     // i
154     angle_tmp_i += ki_angular*diff_theta;
155     if (angle_tmp_i >= rot_vel_i ) angle_tmp_i=rot_vel_i;
156     else if (angle_tmp_i <= -rot_vel_i) angle_tmp_i=-rot_vel_i;
157
158     double angle_tmp = angle_tmp_p + angle_tmp_i;
159     if (angle_tmp >= rot_vel ) angle_tmp=rot_vel;
160     else if (angle_tmp <= -rot_vel) angle_tmp=-rot_vel;
161
162     twist.angular.z = angle_tmp;
163
164     //maximum velosity
165     double dist_tmp = kp_linear * diff_dist;
166     if (dist_tmp >= linear_vel ) dist_tmp=linear_vel;
167     else if (dist_tmp <= -linear_vel) dist_tmp=-linear_vel;
168     twist.linear.x = dist_tmp;
169
170     twist_pub.publish(twist);
171     ros::spinOnce();
172     loop_rate.sleep();
173 }
174 twist.angular.z = 0;
175 twist.linear.x = 0;
176 twist_pub.publish(twist);
177 ros::spinOnce();
178 ROS_INFO("Arrived at WP %d",no);
179 sleep(1); // [s]
180 }
181
182 void gotoWaypoint(int no)
183 {
184     gotoPosition(no);
185     gotoDirection(no);
186 }
187
188
189
190 int main(int argc, char **argv)
191 {
192     ROS_INFO("Start");
193
194     //magic, don't think about here.
195     ros::init(argc, argv, "happy_navi");
196     ros::NodeHandle nh;
197     ros::Subscriber odom_sub = nh.subscribe("odom", 1, odomCallback);
198     twist_pub = nh.advertise<geometry_msgs::Twist>("cmd_vel",1000);
199     pose2d_pub = nh.advertise<geometry_msgs::Pose2D>("roomba_pose2d", 1);

```

```
200     ros::Rate rate(10);
201
202     Arm arm(&nh);
203
204     //initializing way point.
205     z axis.
206     int next_wp = 0; // Next waypoint
207     int next_ap = 0; // Next armpoint
208     sleep(1);
209     //initializing arm position
210     arm.armPos(armpose[11]);
211     while(ros::ok() && arm.moveCheck()){
212         ros::spinOnce();
213         arm.cycle();
214     }
215
216     //main while
217     while (ros::ok() && waypoint[next_wp].x != qqq) {
218         ros::spinOnce();
219         //way point running
220         ROS_INFO("Go to WP %d",next_wp);
221         gotoWaypoint(next_wp);
222         arm.armPos(armpose[next_ap]);
223
224         while(ros::ok() && arm.moveCheck()){
225             ros::spinOnce();
226             arm.cycle();
227         }
228         next_wp++;
229         next_ap++;
230     }
231
232     twist.angular.z = 0;
233     twist.linear.x = 0;
234     twist_pub.publish(twist);
235     ROS_INFO("Mission complete!");
236     sleep(3); // [s]
237     return 0;
238 }
```