2024
春号

百萬石

Hyakumangoku Vol.62

Published by MMM

部長挨拶

gae

gae@mma.club.uec.ac.jp

「百萬石」をお手に取っていただき，ありがとうございます．弊誌 (以下，部誌) は，電気通信大学の公認サークル MMA が新歓期 (春号) と弊学が開催する調布祭時期 (秋号) の年 2 回発行・頒布しているものです．MMA とは，”Microcomputer Making Association”の頭文字に由来しており，1975 年に創立されたサークルです．現在では計算機に関連することを中心に，ネットワークやセキュリティなどの分野でも活動しています．部誌には，MMA の部員が各々に行なっている活動を記事にしたものが掲載されています．内容は多岐に渡り，高度なプログラミングの話から旅行記など，MMA の活動の自由さが感じられるものとなっております．

さて，この「春号」は新歓期に頒布されるものです．昨年度に引き続き，今年度も MMA は新歓に対面で参加するため，ありがたいことに直接部誌をお渡しできます．オンライン参加時代に作った，これまでの歴代の部誌がすべて pdf で読めるようになっている，部員が作成したサイトもあります．先輩方の歴史とともにお楽しみいただければと思います．

今回は春号ということもあり，ここからは新入生向けに話します．技術系サークルにはすごい人たちが集まっているというイメージがあるでしょう．確かにすごい人たちはいますが，しかし**未経験者も多く入部しています．**全くの未経験で入部してから学び始める人も多くいます．私も 1 年生のときには完全な初心者でした．このサークルには多くの経験者がいるため，初心者は経験者に教わりながら学ぶことができます．**コンピュータを使っていれば，活動内容は完全に自由です．**部内でもさまざまな活動が行われているため，質問すれば誰かしらが答えてくれます．

しかし，「好きなことややりたいことができる」ということは「やることを探さなくてはならない」ということでもあります．やりたいことがない人は，なにをやりたいのか見つけることから始めてみましょう．昨年度も，入部したのはいいものの，何をすればいいかわからない部員が幽霊と化していくことは珍しくありませんでした．

そこで我々は**新入生講習会**を用意しました．入部していない方も対象のものもあります．この講習会の目的は単なる勉強だけではありません．先輩との接点を持ち，雑談など様々な会話をしていく中で，自分の興味はどこにあるのかを探し，**やりたいことを見つける**ところにもあります．5 月以降は，対象を部員に絞って，部員がやりたいことをやるための環境を整えていきます．より専門的な講習会も行い，今後のプログラミングの活動のための基礎を固めていきます．是非参加して，今後の自分のやりたいことや大学に役立つことなど，いろいろなことを見つけて学んでいってください．

ちなみに，もう少しサークル紹介をしておくと，MMA は**花火打ち上げサークル**です．（は？）もう少し詳しく説明しておくと，MMA は毎年合宿を行っています．その合宿のメインイベントとして花火を打ち上げます．そのために，**部員が免許を取り**，**打ち上げの準備をして**，**打ち上げを行う**ということを行っています．昨年実際に打ち上げた花火が下の写真です．今年度も行う予定なので，花火を打ち上げてみたいという新入生の方はぜひ入部してみてください．先輩打ち上げ師がしっかり教えてくれます．



最後に，MMA は部員が自由に活動できるサークルです．せっかく電通大に入学した大学 1 年生の皆さん．プログラミングを楽しく学んでみませんか？ MMA に入るのに必要なのはやる気だけです．一緒に頑張っていきましょう．

2024 年 4 月吉日

百萬石 Vol.62

Hyakumangoku Vol.62

2024 Spring

MMA

目次

第 1 章　まだ LaTeX 使ってるの？ gae 1
1.1 はじめに 1
1.2 Typst って？ 1
1.3 Typst の特徴 1
1.4 Typst 言語 2
1.5 まとめ 3

第 2 章　パスワード設定のあれこれ udon 7
2.1 はじめに 7
2.2 パスワード設定において意識していること 8
2.3 おわりに 11

第 1 章

まだ LaTeX 使ってるの？

gae

1.1 はじめに

こんにちは．私の記事を読んでくださりありがとうございます．2024 年度の部長の gae です．部長挨拶以外にも部誌を書くのは 2 回目でまだ慣れていませんが，読んでくだされば幸いです．

1.2 Typst って？

今回は **Typst** について書こうと思います．Typst とは，マークアップベースの新しい組版システムです．LaTeX とは異なり，コンパイルが非常に高速です．マークダウンのように高速にコンパイルし，pdf を出力することができます．公式サイトには書いてありますが，Typst は，科学分野向けで LaTeX への不満から生まれたものとされています．数式や相互参照などの機能も備えており，LaTeX に近い機能を持っています．

1.3 Typst の特徴

- かなり新しい組版システムである
  - 2019 年に開発が始まった．
  - オープンソース化されたのは 2024 年 3 月．
- Rust 製である．
- 公式ドキュメントがかなり充実している．
- 独自の構文を用いることができる．

今回は特に独自構文 (以下 Typst 言語とする) について書こうと思います．

1.4 Typst 言語

組版用の言語では，どの程度簡潔に軽量に書けるようにするのか，どの程度複雑な内容も記述できるようにするのかは，トレードオフの関係にあることが一般的です．以下の例を見てみましょう．

```
= 見出し1

これは段落です．空行は改段落を表します．
// 2つのスラッシュから始まる行はコメントです．このテキストは出力に現れません
    ．

== 見出し2

テキストは _強調 (emphasis)_ することもできれば，
*強調 (strong emphasis)* することもできます．

- これは箇条書きです．
  - インデントは箇条書きのネストを表します．
- 先頭を ‘+‘ とすれば番号付き箇条書きとなります．
```

これは Typst 言語で書かれたシンプルな文書です．Markdown にかなり似ています．次に以下の例を見てみましょう．

```
#heading(level: 1)[見出し1]

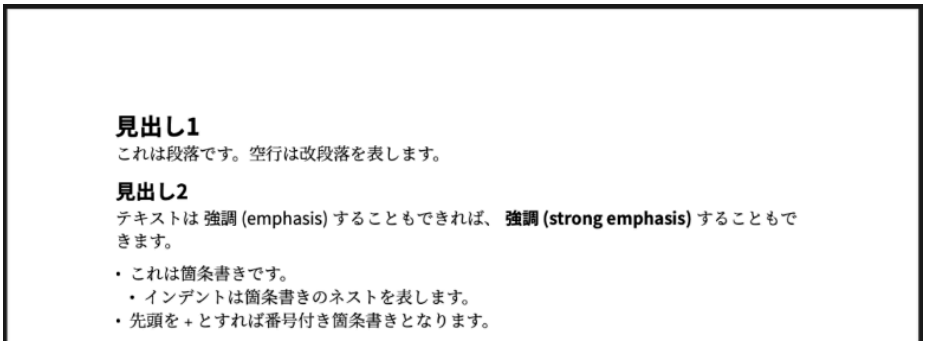
これは段落です．空行は改段落を表します．

#heading(level: 2)[見出し2]

テキストは #emph[強調 (emphasis)] することもできれば，
#strong[強調 (strong emphasis)] することもできます．

#list(
    [
    これは箇条書きです．
    #list[インデントは箇条書きのネストを表します．]
    ],
    [
    先頭を #raw("+") とすれば番号付き箇条書きとなります．
    ],
).
```

この例は，先ほどとは異なりますが，ドキュメントには `embed a code expression into markup`と書かれているように，関数呼び出しなどの指揮をマークアップ言語内に埋め込むための記法です．書き方はどちらも異なりますが，出力結果はどちらも以下のようになります．



1.5 まとめ

私も実際に最近この Typst をしり，使ってみました．下のコードは私の作成した Typst のテンプレートで，電気通信大学のレポート等に今後使用する予定です．また， LaTeX と同様に，Typst にもテンプレートが存在し，それを利用することで，簡単に文書を作成することができます．自分でテンプレートを作成することもできますが，既存のテンプレートを利用するこで，簡単に文書を作成することができます．私は既存のものから設定を変更し，自分の好みに合わせています．

ちなみにこのサンプルコンパイル時間は 1 秒もかかりません．画像やソースコード，数式，参考文献なども入っていますが，しっかりと高速でコンパイルすることができます．皆さんもぜひ Typst を使ってみてください．VSCode にもすでに拡張機能があったりブラウザ上で実行できるサイトも準備されているので，公式のリポジトリやドキュメントを見ながら，ぜひ始めてみてください．快適なレポート生活を送りましょう．ここまで読んでいただきありがとうございました．

ソースコード 1.1 template.typ

```
#import "@preview/codelst:2.0.1": sourcecode // sourcecode

#let textL = 1.5em
#let textM = 1.3em
#let fontSerif = ("Toppan Bunkyu Mincho")
#let fontSan = ("HackgenNerd")

#let project(title: "", authors: "", course:"", body) = {

    // Heading
    set heading(numbering: "1.1")
    show heading: set text(font: fontSerif, weight: "medium", lang: "ja")
    show heading: it => {it; par(text(size:0pt, ""))}
    show heading.where(level: 1): it => {
        set text(size: 1.1em)
        pad(top: 0.7em, bottom: 0.7em)[#it]
        par(text(size:0pt, ""))
    }
    show heading.where(level: 2): it => {
        pad(top: 0.5em, bottom: 0.6em, it)
        par(text(size:0pt, ""))
    }
```

```
    // Figure
    show figure: it => pad(y: 1em, it)
    show figure.caption: it => text(it, lang: "ja")

    // Table
    show figure.where(kind: table):set figure.caption(position: top)
    set table.hline(stroke: none)

    // Outline
    show outline.entry: set text(font: fontSerif, lang: "ja")
    show outline.entry.where(
        level: 1
    ): it => {
        v(0.2em)
        set text(weight: "semibold")
        it
    }

    // Set the document's basic properties.
    set document(author: authors.map(a => a.name), title: title)
    set page(numbering: "1", number-align: center)
    set text(font: fontSerif, lang: "jp")
    set par(justify: true, first-line-indent: 1em)
    show math.equation: set text(weight: 400)

    // Title row.
    let date = datetime.today()
    align(center)[
        #block(text(weight: 500, 1.5em, title))
        #block(text(weight: 400, 1.1em, course))
        #v(1em, weak: true)
        #date.year() 年 #date.month() 月 #date.day() 日
    ]

    // Author information.
    pad(
        // bottom: 0.5em,
        grid(
            columns: (1fr,) * calc.min(3, authors.len()),
            gutter: 1em,
            ..authors.map(author => align(center)[
                    #author.name (#author.number)\
                #author.affiliation
            ]),
        ),
    )
    set text(font: fontSerif, size: 10pt, weight: "light", lang: "ja")
    body
}
```

ソースコード 1.2　main.typ

```
#import "./../template.typ": *

#show: project.with(
    title: "情報通信実験",
    course: "情報通信工学実験A",
    authors: (
```

```
        (number: 2210999, name: "gae", affiliation: "情報通信工学プログラム"),
    )
)

= いろいろな例

ここでは，論文の表紙および本体の記述方法について述べる．
ああああああああああああああああああああああああああああああああああああああああ
    ああああああああああああああああああああああああああああああああああああああ
    あああああああああああああああああああああああああ

== 画像

画像を挿入する例を @figure_image に示す．

#figure(
    image("./../images/mma-logo.jpg", width: 80%),
    caption: [MMAロゴ],
)<figure_image>


== 表

表を描画する例を @figure_table に示す．

#figure(
    caption: [表の例],
    table(
        columns: 2,
        [*西暦*], [*和暦*],
        [2023年], [令和5年],
        [2024年], [令和6年],
    ),
) <figure_table>

== 数式

$ integral_(1)^(oo) f(x) dif x $

$alpha, beta, gamma$
$Alpha, Beta, Gamma$

#sourcecode[```python
def hello():
    print("Hello, world!")

if __name__ == "__main__":
    hello()
```]

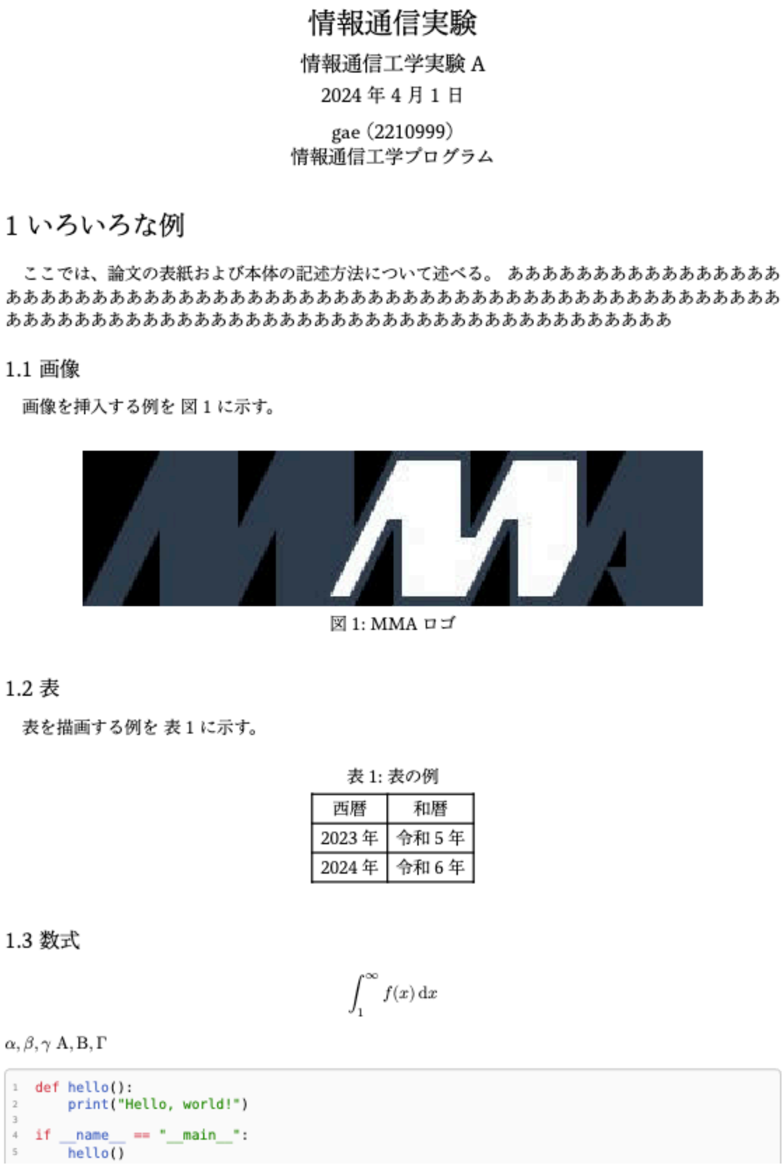
@repo-typst-coins-thesis

== 脚注

本テンプレートはTypst #footnote[https://typst.app/] 用に作られている．

#[
    #bibliography(title: "参考文
       献", full: true, style:"ieee", "./../bib.yaml")
```
```

```
]
```



情報通信実験

情報通信工学実験 A

2024 年 4 月 1 日

gae（2210999）
情報通信工学プログラム

1 いろいろな例

ここでは、論文の表紙および本体の記述方法について述べる。 ああ

1.1 画像

画像を挿入する例を 図 1 に示す。

図 1: MMA ロゴ

1.2 表

表を描画する例を 表 1 に示す。

表 1: 表の例

西暦	和暦
2023 年	令和 5 年
2024 年	令和 6 年

1.3 数式

$$\int_1^\infty f(x)\,\mathrm{d}x$$

α, β, γ $\mathrm{A}, \mathrm{B}, \Gamma$

```
def hello():
    print("Hello, world!")

if __name__ == "__main__":
    hello()
```

図 1.1　サンプル出力結果

第 2 章

パスワード設定のあれこれ

udon

2.1 はじめに

お疲れ様です。Udon です。

今回部誌を書くつもりはなかったのですが、なんか記事が少なくて寂しいどころではなかったので締切直前でもさっと書ける簡単な話題を扱おうかなと思います。

さて、これを読んでいる皆さんは多くが情報系を学んでいる人だと思うのですが、情報を扱ううえで大切なことはなんでしょうか。データが改ざんされていない完全性、いつでもデータを利用できる可用性、実行時間をできるだけ短縮できるような効率性、どれも大切なことです。しかし、やはり忘れてはいけないのがデータが外部に漏れないようにする「機密性」です。これが守られていないと、個人情報が流出して勝手に自分の名前を名乗るやつが現れたり、クレジットカードの情報が漏れだしたりでもしたら自分のお金が勝手に使われるなんて最悪の事態にも発展しかねません。

データの機密性を守るうえでとても一般的な手段として、「パスワード」というものがあります。これは事前に ID とともにサーバに登録され (そのままではなく、ハッシュ化された形で保存されることが多いです)、ユーザが ID とともに入力することで一致しているかのチェックを行い、ユーザ認証を遂行します。

パスワードは簡単かつ自由に設定でき、設定するだけである程度のセキュリティ効果を発揮することができます。しかし、「バレてしまうと終わり」だったり「総当たり攻撃で突破されかねない」というような弱点も抱えています。そのため、パスワードは「複雑」で「長い」だけでなく「できるだけ多くの種類の文字が使われている」必要があるでしょう。でも、複雑なパスワードはとても覚えづらく、忘れてしまいやすいという欠点も抱えています。

今回はこうしたパスワードの問題とどう付き合っていくかについて自分が意識していることを紹介しようかなと思います。

2.2 パスワード設定において意識していること

2.2.1 パスワードマネージャーの利用

現代は便利になったもので、iPhone などの携帯電話にはパスワードマネージャーが搭載されています。これはサイトなどとパスワードを関連付けてローカルに保存しておき、必要に応じて呼び出す、といった機能を持つものです。ローカルに保存しているため機器そのものを盗まれない限り流出の危険が少ないでしょう。場合によってはパスワードを他のユーザとシェアできるらしいですが、そういった設定は必要ないならやらないほうが望ましいと思われます。

Edge や Chrome などの Web ブラウザにも搭載されており、Edge は同じ Microsoft アカウントを使っている機器同士で同期ができるようです。どうやら Microsoft アカウントに紐づくデータベースにパスワードが保存されているみたいですね。

この機能の利点はなんといっても「長く複雑なパスワードでも一度登録してしまえば自動で入力してくれる」ということでしょう。これによって強力なパスワードを設定しつつも、入力や記憶の手間を省くことができます。この機能は積極的に利用すべきだと思います。

特に、クレジットカード情報や個人情報を登録したサイトには強力なパスワードを設定すべきだと思います。

2.2.2 長さ

長さなのですが、サイトによっては長さ制限があるものがあります。その場合は、長さ制限ギリギリのものにしています。理由としては、長ければ長いほどパスワードの考えられるパターンが加速度的に増えるからです。アルファベット (大文字と小文字) と数字を合わせたら 62 文字あるのですが、5 文字なら $62^5 = 916132832$ で大体 9 億通りで、列挙にさほど時間はかからないのですが、20 文字に延ばすだけで $62^20 = 704423425546998022968330264616370176$ で大体 7000 澗通りくらいになりました。これを総当たりで突破するには多大な計算量を要求することになるので、攻撃者側もやるメリットが薄くなるでしょう。これに記号も併せると、よりパターンは増え、強力なパスワードになります。

2.2.3 用いる文字

サイトによって使える文字の種類は異なっていますが、これも最大限使うようにしています。数字だけだとパターン数は 10^n(n は文字数) 程度ですが、ここに大文字小文字と記号 10 種類を加えるだけでパターン数は 72^n に跳ね上がります。

また、用いる文字の繰り返しはできるだけ避けると良いと思います。無作為に文字を選ぶことで、安全性は高まると思います。

- 良い例：`k*HngPjoe_10934`
- 悪い例：`aaaaaaaaa_11111`

2.2.4 使いまわし

これは絶対にやめた方が良いと思っています。使いまわせばそれだけ流出の危険が倍増するといっても過言ではありません。パスワードマネージャーに頼るなどして、毎回異なる強力なパスワードを設定したほうがいいと思います。

2.2.5 メモ

パスワードマネージャーがあるとはいえ、メモも残しておいたほうが万一の事態に対応できるでしょう。PC のローカルファイルにメモしておいたり、スマホのメモ機能を用いてメモしたりすると良いと思います。万一の事態に備え、そういったメモにはパスワードのロックをかけておくと良いです。そのパスワードは自力で覚えることになりますが他の全てのパスワードを守るものなので、強力なパスワードであり、かつ機器本体をロックするものとは異なるパスワードであることが求められます。しかし、それさえ覚えておけば他の全てを守れるので、ここは頑張って覚えてもいいのかなと思います。

誰かはパスワードを紙にメモし、金庫に入れていると言っていました。これも金庫の鍵と位置を秘密にすればよいので、良い手段でしょう。

2.2.6 ランダム性

パスワードマネージャーに提案されるパスワードはランダムに設定されたものなのですが、例えば iPhone のものは以下のような形式に固定されているようです。

kl2hM1-A4bnV2-jatbmR[*1]

これは英数字計 62 文字を 18 字並べ、6 文字で区切ってハイフンを入れるといった形式になっています。これでもパターン数は 62^{18} で結構多いのですが、記号がハイフンだけでその位置も固定されているので、不安も残りますね。そういう時はパスワードを自動生成してくれる Web ツールを使うのも手でしょう。こういったツールは数多く公開されていますし、簡単にプログラミングして作ることもできます。

ただ、以下のように、「パスワードマネージャーに頼らず自分で覚えておくべきパスワード」を設定する必要がある時もあるでしょう。

- パスワードのメモを守るパスワード
- PC にログインする時など、よく手動入力するパスワード
- 様々な機器で使うパスワード

こういう時、覚えやすくかつ規則がバレづらいパスワードを設定する必要があるでしょう。規則がバレやすいというのは以下のようなものです。

[*1] 適当に決めたやつなのでどこかのパスワードではないです

- Udon2003：ユザネ + 誕生年。安直なので適当に打ってても当てられる可能性がある。
- dentsudai：単語のみ。辞書に載っているような単語の組み合わせは「辞書攻撃」というものによって突破されやすい。
- poepoepoepoe：繰り返し。長さは確保できているがこの例だと「poe が 4 連続」という情報だけで突破可能である。 ぽえぽえ〜

では、どのように覚えやすさと不規則性を両立させるか。自分が意識していることには以下のようなものがあります。

今聴いている曲から取る

自分はよくシャッフル再生で音楽を聴くので、設定時に聴いている曲から連想される言葉をパスワードに用います。曲名やアーティスト名だと安直なので、歌詞の一部などを採用することが多いです。

採用する言葉を改造する

例えばなんですが、「bokunomononinaruwakenaika」という一節[*2]を使うとしましょう。これだけだと日本語なのでちょっと不安です。そこで、「前半は子音だけ、後半は母音だけにする」という形で改造してみます。すると「bknmnnauaeaia」となり、ちょっと複雑になりました。少なくとも辞書に載っている言葉ではないですね。思いだすときには、使った言葉とどこが子音になっててどこが母音になっているのかという情報を使うだけでいいので、かなり記憶コストはおさえられると思います。

乱数を入れる

適当にテンキーを叩いてみたり、現在時刻から決めたりした乱数を入れると数字を取り入れることができ、パターン数を 10^n 倍できます (n は数字の数)。例えば今日は 4/2 なので 0402 を入れてやると例のパスワードは「bknmnnauaeaia0402」になりました。

工夫する

例えば、偶数文字目のアルファベットは大文字にしたり、「a」を「@」に置き換えるなどの工夫をしてやると、覚えやすさを保ったまま大文字や記号を取り入れることができ、更に強力になります。

これによって例のパスワードは「bKnMnN@U@E@I@0402」になりました。

思いだすための要素を整理しておく

さて、このパスワードを思い出すための要素は以下の通りです。

- ベースの言葉「僕のものになるわけないか」
- 「僕のものに」が子音
- 「なるわけないか」が母音

[*2] backnumber の曲から取りました

- 乱数 0402 とそれを入れる位置
- 偶数番目のアルファベットは大文字
- a を@にする

この要素を覚えていくことで、ちょっと思いだすのに時間はかかるかもしれませんが、それなりに複雑なパスワードを覚えやすくなるような気がします。

2.3 おわりに

情報技術に携わる人間として、パスワードはしっかりと設定したほうがいいと思います。上記で紹介したことはあくまで Udon が意識していることなので、他にもいいやり方はあると思います。自分に合った方法で安全性と使いやすさを両立させておくといいかなと思います。

くれぐれもパスワードを破られて、大切なデータが流出したり悪用されたりすることがないようにしましょう。

製本直前に急いで書いたので適当な文章になってしまっていますが、最後まで読んでいただきありがとうございます。

百萬石 2024 -春- © 電気通信大学 MMA

2024 年 4 月 3 日　初版第一刷発行　【本書の無断転載を禁ず】

著　者　gae, Udon
表　紙　Udon
編集者　Udon
発行者　電気通信大学 MMA
発行所　電気通信大学 MMA 部室
〒182−8585 東京都調布市調布ヶ丘 1−5−1 サークル会館 208 号室
https://www.mma.club.uec.ac.jp/
印刷所　電気通信大学 MMA 部室
製本所　電気通信大学 MMA 部室


電 気 通 信 大 学