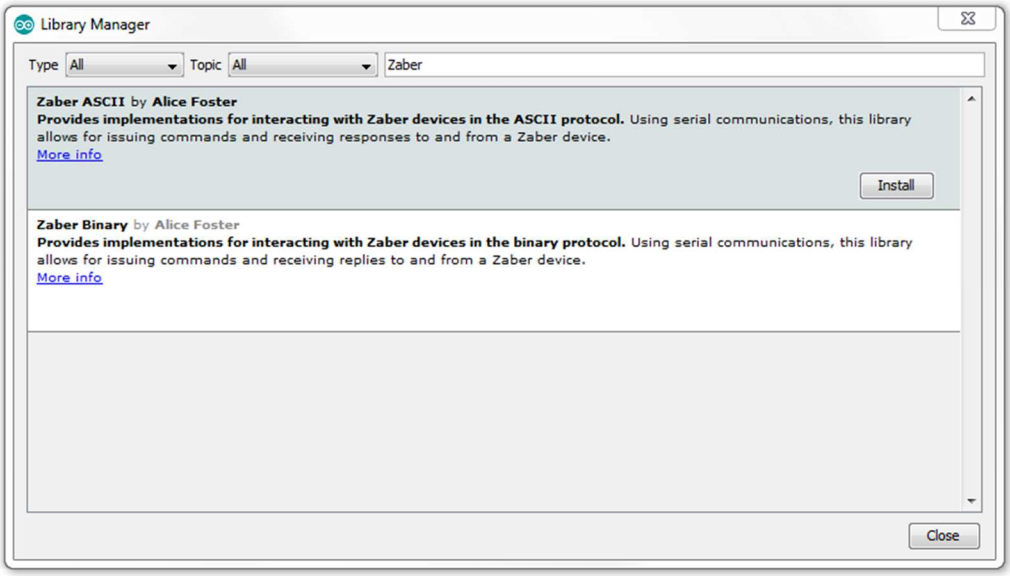
# Zaber ASCII ライブラリ(アルデューノ用)

## 目次


1. アルデューノ用 Zaber ライブラリのダウンロード
2. 初心者向け詳細ガイド
    - ユーザ Zaber デバイスとアルデューノボードとの接続
    - コード作成 – アルデューノ IDE
    - 簡単な ASCII 原点復帰サンプル
    - 中級 ASCII プログラミング例
    - シリアルモニターを使ったユーザコードの監視
    - 返答の翻訳
    - シールドやセンサーの追加
3. 自作のプログラムが動作しない!
4. 追加サンプルプログラム
5. 付録 A: しっかりした I/O インターフェース活用
6. 付録 B: Zaber の X-シリーズ シールド無しで開始
7. 関連資料


## アルデューノ用 Zaber ライブラリのダウンロード

1. Arduino IDE のダウンロードとインストトール
2. Arduino IDE オープンとスケッチ – >ライブラリを含める – >ライブラリの管理... への案内
3. 表示されたウィンドウ内で、[検索を絞り込む...]ボックス内で "Zaber"を検索します。
4. 表示されている各ライブラリをクリックしてから、[インストール]ボタンをクリックします。.




## 詳細な初心者向けガイド

このチュートリアルでは、簡単な機能例の設定の詳細について説明します。 ユーザの Arduino で動作する最小の例を得ることに関する特別コースについては、クイックスタートガイドにてご覧いただけます。

## Zaber デバイスを Arduino ボードへ接続

Arduino のボードは小さなコンピュータですが、一度に実行できるプログラムは 1 つだけです。 これにより、プロトタイピング、テスト、またはこの場合はモーターとのやり取りのための安価で便利なツールとなります。

Zaber デバイスはシリアル通信を介して Arduino ボードと通信します。 つまり、Arduino を Zaber デバイスに接続するには RS-232 シリアル通信ポートとケーブルが必要です。 Zaber は X-Series Shield を推奨しています。これにより、Zaber デバイスとデバッグ用シリアルモニターを同時に通信できます。 他のデバイスに追加のシリアルポートを使用したい場合は、Arduino Mega のような大規模プラットフォームをご検討下さい。

この初心者ガイドの説明対象商品:

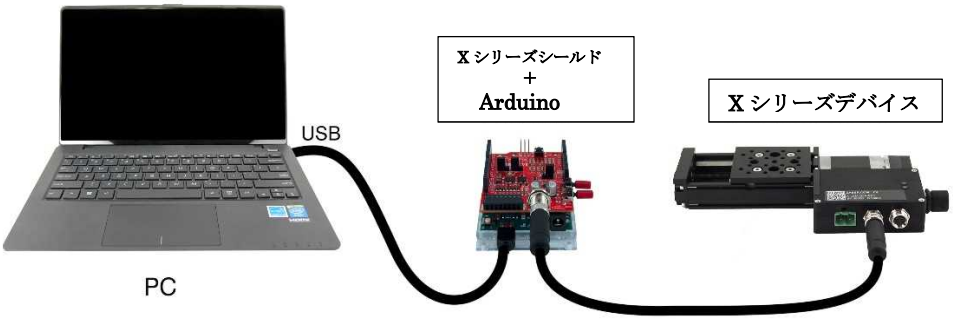
- Zaber デバイス(ステージ)
- Arduino Uno Rev 3 または Mega Rev 3 ボード
- Arduino を PC に接続するための USB ケーブル(これは Arduino に付属しているはずです)
- Arduino 用のXシリーズシールド
- Zaber デバイスと X シリーズシールドを接続するための X-DC データケーブル
- A シリーズおよび T シリーズデバイスには、代わりにそれぞれ S-XDC または T-XDC が必要です。

X シリーズシールドを使用すると、Zaber X シリーズデバイスから Arduino に電源を供給できます。これは、プログラムを終了してコンピュータに接続しなくても実行したい場合に役立ちます。 しかし現時点では、USB 経由で Arduino に電力を供給するのが最善です。

- 電源ジャンパが "Vin / USB"の位置にあることを確認してください。

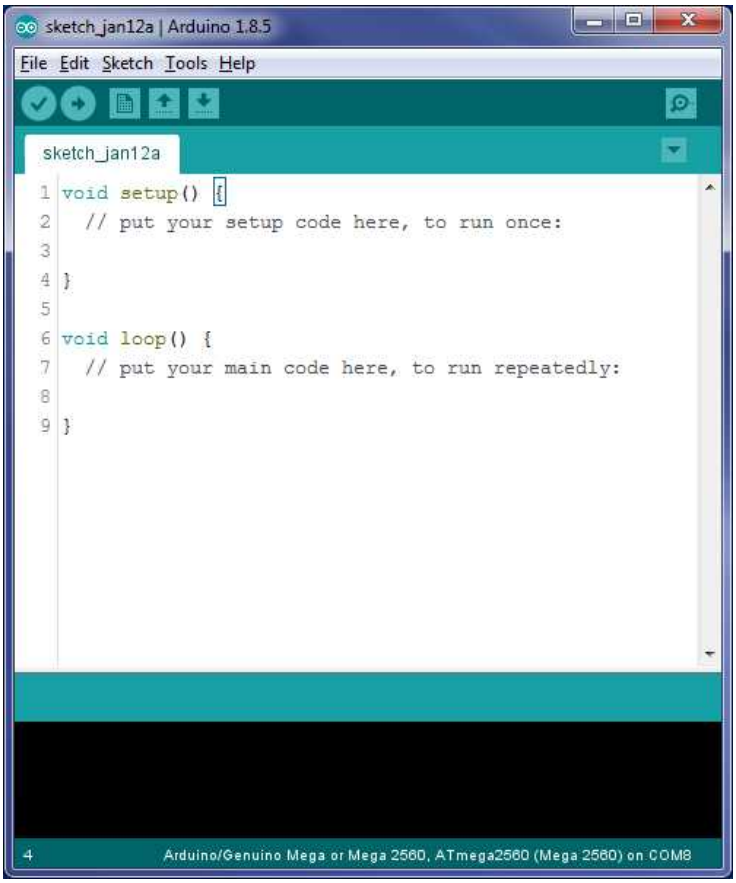
Zaber の X-シリーズ Shield は Arduino と通信するために I2C を使用します。 ほとんどの場合、初期値のアドレス AA で機能します。 I2C も使用する他のシールドを使用することを計画している場合は、必ず各シールドに固有の I2C アドレスを設定するために、 必要なすべての Shield とセンサーの追加をご考慮ください。

1. X シリーズシールド(X-AS01)を Arduino に接続して下さい。
2. 正しいデータケーブルで Zaber デバイスを X シリーズシールドに接続します。
3. Zaber デバイスに電源を入れます。
4. USB ケーブルで Arduino をコンピュータに接続して下さい。 X シリーズシールドの "ON" LED が点灯します。.



## コードを書く – Arduino IDE

Arduino プログラムは、Arduino IDE を使用してコンパイルされ、Arduino ボードにアップロードされます。こちらからダウンロードできます。 Arduino IDE をインストールしてコンピュータに開くと、次のような画面が表示されるはずです。




これがすべての C ++コードが Arduino ボード用に書かれコンパイルされている場所です。 setup()関数に含まれるコードは、Arduino が最初にプログラムを起動したときに 1 回実行されます。 Arduino ボードに電源が入っていてプログラムが実行を停止していない限り、loop()関数内にあるコードが繰り返し呼び出されます。

Arduino プログラムを実行しようとする前に、Arduino IDE にどのタイプの Arduino があり、どのポートがコンピューターに接続されているかを伝えなければなりません。 Arduino IDE のドキュメントを参照してください。

---

## 簡単な ASCII ホーミング（原点復帰）の例

このセクションでは、概念を説明しながら、簡単なプログラムを段階的に作成するプロセスを説明します。完全なプログラムに直接ジャンプしたい場合は、メニューから File – > Examples – > Zaber ASCII の順に移動して、Arduino IDE で提供されている例の 1 つを開くことができます。そこには、Basic Homing（基本的原点復帰）動作の例が載っています。

手始めに、簡単なプログラムを書いてみましょう:Arduino の電源が入ったら、接続されている Zaber デバイスをホームポジションに送ります。 このコマンドは一度だけ発行すればよく、繰り返し発行する必要はないので、この場合は setup()関数を使用して、loop()関数を空のままにすることができます。

**Arduino 用に Zaber のライブラリを初期化する o**

デバイスがバイナリモードの場合は、最初に ASCII に切り替える必要があります。 Zaber コンソールを使用して、デバイスの動作モードを変更できます。 その[オプション]タブの下には、すべてのデバイスを ASCII プロトコルとバイナリプロトコルの間ですばやく変更するためのボタンがあります。 この例では、デバイスは ASCII モードであると仮定します。

最初にすべきことはあなたのファイルの一番上に ZaberAscii ライブラリをインポートすることです。

**よくある質問**

*ライブラリーとはなに？*

ライブラリーは、一緒に配布されている事前定義関数のグループです。 ライブラリをインポートすると、その関数を自分のプログラムに利用することができます。

*Function(関数)とはなに?*

関数は、名前が付けられたコードの短いセクションです。 同じコードが何度も重複するのを避けるために、その名前を使用してプログラム内の複数の場所から実行できます。

*どんな機能が利用可能かはどうすれば分かりますか？*

以下のコード例では、 "ZaberAscii.h"はライブラリと一緒に配布されているファイルを参照しています。 このファイルを開くと、関数の宣言を確認することができます。 通常、ライブラリにはこれらの.h ファイルが含まれており、他の参照ドキュメントが利用できない場合に参照できます。このライブラリには、個別の関数参照ドキュメントは含まれていませんので、ZaberShield.h と ZaberAscii.h を参照してください。 Windows PC では、これらのファイルの一般的なインストール場所は Documents ¥ Arduino ¥ libraries ¥ ZaberAscii です。 Arduino が提供する関数のリファレンスはここにあります。

Arduino IDE で新しいファイルを作成し、この行を先頭に追加します。

```
#include <ZaberAscii.h>
```

**インスタンスを作成する**

さて、ZaberAscii クラスのインスタンスを初期化する必要があります。 その際には、Serial クラスをコンストラクターに渡す必要があります。 シリアルクラスはシリアルポートを通して Zaber デバイスとの通信を制御します。 すべての Arduino ボードには少なくとも 1 つのシリアルポートがあるので、すべての Arduino ボードには基本的なシリアルクラスがあります。ただし、Arduino ボードに複数のシリアルポートがある場合は、クラス Serial1、Serial2、Serial3 を追加で利用でき、Zaber 以外のデバイスとの同時制御、またはデバッグ印刷が可能になります。Zaber X-Series Shield では、ライブラリで提供されている ZaberShield という別のシリアルクラスを使用する必要があります。

まず、ZaberShield クラスのインスタンスを定義して初期化します。 このインスタンスには、ボード上のアドレスジャンパ設定に関する情報を提供する必要があります。

```
#include <ZaberAscii.h>

ZaberShield shield(ZABERSHIELD_ADDRESS_AA);
```

**よくある質問**

*インスタンスとは何ですか。なぜ必要なのですか？*

私たちのネームシールドは、使用するアドレスを設定する数値定数を使用して初期化された ZaberShield クラスのインスタンスです。 このインスタンスは、ZaberShield クラスのデータのコピーを保存するために使用されるメモリのセクションであり、それによってクラスのすべての機能を利用することができます。 ZaberShield の他のインスタンスを作成し、それぞれに異なる名前を付けることで、それぞれが独自の内部状態を個別に追跡することができます。

*クラスとは？*

クラスとは、ゼロ個以上のデータとそのデータを処理する関数の集まりです。 クラスは、振る舞いとデータをグループ化することによって、概念的にコードを実世界のオブジェクトに似せるものであり、オブジェクト指向プログラミングの中核的概念です。

*コンストラクタとは何ですか？*

コンストラクタは、インスタンスを作成したいときに呼び出されるメソッドです。 コンストラクタに渡されるパラメータは、インスタンスを作成するものです。 この例では、コンストラクタへのパラメータとして ZABERSHIELD_ADDRESS_AA を使用して shield を構築します。

*初期化とはどういう意味ですか？*

初期化とは、インスタンスの初期状態を設定することです。 数値データの場合、これは最初の値を代入するだけです。 クラスインスタンスの場合、これはコンストラクタを呼び出すことを意味します。コンストラクタは、クラスの内部データの初期化を担当します。

*グローバル変数とは？*

グローバル変数は、プログラム内のどこからでもアクセスできる変数です。これは、すべてのネストした関数がアクセスできるようにするために、ファイルの先頭で宣言された最高の有効範囲を持ちます。 上記の例では、shield はグローバル変数です。 グローバルではない変数の例としては、クラス内または関数内で宣言されているものがあります。

上記のコードは、shield という名前のグローバル変数を、ポートアドレス 0x90 を使用する ZaberShield シリアルポートクラスのインスタンスになるように設定します。 ZABERSHIELD_ADDRESS_AA という名前は、この目的のためにライブラリーに定義されている 4 つの定数のうちの 1 つです（名前が AA、AB、BA、および BB で終わる場合）。

これらの定数名の接尾辞の最初の文字は AD1 のジャンパ位置に対応し、2 番目の文字は AD0 のジャンパ位置に対応します。 シールドのジャンパ設定と一致する定数を使用する必要があります。

ポートアドレスは Zaber シールドのジャンパを使って設定できます。他の目的のために与えられたアドレスを使いたい別のシールドとの衝突を避けられるようにアドレスは設定可能です

次に、ZaberAscii クラスのインスタンスを za という名前のグローバル変数として定義し、上で作成したシールドクラスを渡します。

以下の増分コード例では、「この行は新しいです」というコメントで変更が強調されています。 これらのコメントをコードに含める必要はありません。

```
#include <ZaberAscii.h>

ZaberShield shield(ZABERSHIELD_ADDRESS_AA);
ZaberAscii za(shield);  // <-- This line is new.
```

**ボーレートの設定**

ZaberAscii インスタンスが初期化されたので、ボーレートを設定してデバイスに合わせることが重要です。 Zaber X シリーズデバイスは、デフォルトで 115200 のボーレートを使用します。setup()関数で、このボーレートを使用して ZaberShield クラスを初期化しましょう。起動時に呼び出されます。

```
#include <ZaberAscii.h>

ZaberShield shield(ZABERSHIELD_ADDRESS_AA);
ZaberAscii za(shield);

void setup() {
    /* Initialize baudrate to 115200, typical for Zaber ASCII devices */  // <-- This
line is new.
    shield.begin(115200);                                          // <-- This line is new.
}
```

**Zaber デバイスにコマンドを発行**

すべてのシリアル通信設定が邪魔にならないように，これでデバイスにコマンドを送信できます。 コマンドは、send(int deviceNumber、String コマンド、long int data)関数を使用して ZaberAscii インスタンスに送信されます。この関数のパラメータは次のとおりです。

- int deviceNumber は Zaber コンソールで確認できますが、新しいデバイスの場合はデフォルトで 1 です。これは、すべて同じシリアルポートに接続されている一連のデバイスの中のデバイスのアドレスです。同じポートに複数のデバイスがある場合は、それぞれ異なる番号を設定する必要があります。さらに、deviceNumber = 0 でコマンドを発行すると、そのコマンドはすべてのデバイスに送信されます。

- 利用可能なさまざまな String コマンドの値は、Zaber ASCII Protocol Manual を見ればわかります。 この例では、 "home"コマンドを使います。
- 最後に、コマンド文字列にデータの送信が必要な場合(たとえば、デバイスが移動する距離など)、データは long int data パラメータを介して渡すことができます。このパラメータはオプションであり、コマンド文字列が追加のデータの送信を必要としない場合は無視できます。

## よくある質問

*パラメータとは何ですか？*

数学関数のようなコード関数は、パラメータ化して結果を返すことができます。パラメータ(引数とも呼ばれる)は、関数を呼び出すときに関数に渡すデータです。関数内部のコードは、何をすべきかを決定するために、またはその内部計算の一部として、そのデータを利用することができます。

ボーレートを設定した直後に、定義済みの home コマンド文字列を使用して、デバイス番号 1 に home コマンドを発行しましょう。 このコマンドでは、データを渡す必要はありません。

```
/*デバイスにホームコマンドを発行 1 */
za.send(1, "home");
```

ほとんどの場合、デバイスにコマンドを送信するときは、その応答を読む必要があります。たとえその応答を何にも使用しなくてもです。返事を読んでいない場合は、返事を読もうとするまで Arduino のシリアルポート受信バッファで待機します。すべての返信を読んでいないと、誤って前のコマンドへの返信を読んだり、受信バッファがオーバーフローして破損した返信を読んだりする可能性があります。receive()関数は一つの返事を読むでしょう。 この例では、関数の戻り値を変数に代入しないことによって応答を読んで無視します。

```
/*エラーをチェックしていなくても、必ず返信を読んでください。 */
za.receive();
```

コードは次のようになります:

```
#include <ZaberAscii.h>

ZaberShield shield(ZABERSHIELD_ADDRESS_AA);
ZaberAscii za(shield);
```

```
void setup() {
    /* Initialize baudrate to 115200, typical for Zaber ASCII devices */
    shield.begin(115200);

    /* Issue a home command to device 1 */                    // <-- This line is new.
    za.send(1, "home");                                       // <-- This line is new.



    /* Always read the reply even if not checking it for errors. */  //<-- This line is new.
    za.receive();                                             // <-- This line is new.
}

void loop() {

}
```

### Arduino ボード上でプログラムをコンパイルして実行する

これでプログラムを実行する準備が整いました。 USB コネクタを介して Arduino ボードを PC に接続し、以下のオプションを設定します。

- [Tools]の中で、正しいポートが選択されていることを確認します（Arduino のボード名はかっこ内のポート番号で表示されます）。
- [ツール]の中で、[ボード]を適切なボードに設定します（これは、ポートを選択するときに括弧で囲まれたボード名と同じになります）。

ここで、アップロードボタン（IDE 上部のチェックマークボタンの横にある右向きの矢印）をクリックして、プログラムをコンパイルして Arduino ボードにアップロードします。 プログラムがアップロードされるとすぐに実行され、Zaber シールドにデバイスが接続されている場合は Zaber デバイスに home コマンドを送信します。

---

## 中級 ASCII プログラム例

ご希望の場合は、このガイドに従って、プログラムの書き方について学ぶことができます。 あるいは、Arduino IDE 内で File - > Examples - > Zaber ASCII の順にナビゲートし、Intermediate_LED の例をクリックして完成したコードを Arduino IDE にロードすることもできます。

おそらくあなたは一晩中働いていて、あなたは現在彼らのホームポジションにいない機器だけをモニターしたいと思うでしょう。 ホームポジションにあるデバイスは現在使用されていませんが、まだ電力を受けているため、電源を供給されているがホームポジションにいないデバイスと区別するための方法が必要です。 Arduino の知識を使用して、この問題を解決するためのかなり簡単なスクリプトを書くことができます。

Zaber Ascii クラスのインスタンスを作成し、setup()関数でボーレートを 115200 に初期化して、以前と同じようにプログラムを起動します。

```
#include <ZaberAscii.h>

ZaberShield shield(ZABERSHIELD_ADDRESS_AA);
ZaberAscii za(shield);

void setup() {
    /* Initialize baudrate to 115200, typical for Zaber ASCII devices */
    shield.begin(115200);
}

void loop() {

}
```

次に、ループ機能におけるデバイスの位置を問い合わせます。 この機能は Arduino の電源が切れるかプログラムが停止するまで継続して実行されます。 このように、デバイスの位置が 0 かどうかを確認することで、デバイスの LED ライトをオンまたはオフにすることができます。 まず、デバイスの位置を求めるリクエストを送信します。

```
#include <ZaberAscii.h>

ZaberShield shield(ZABERSHIELD_ADDRESS_AA);
ZaberAscii za(shield);

void setup() {
    /* Initialize baudrate to 115200, typical for Zaber ASCII devices */
    shield.begin(115200);
}

void loop() {
    /* Issue a get position command to the device */          // <-- This line is new.
    za.send(1, "get pos");                                     // <-- This line is new.
}
```

次に、レスポンスを受け取って分析します。 これは reply 構造体を返す receive 関数を使って実行できます。 Zaber ASCII ライブラリからの応答は、struct reply の形式を取ります。 ZaberAscii.h で reply 構造体の宣言を見ることができます。 回答の詳細については、回答の解釈のセクションを参照してください。

## よ く あ る 質 問

*構造体とは何ですか？*

構造体(「構造体」の略)は、クラスのように複数のデータを格納するためのコンテナです。 クラスとは異なり、構造体は通常関数を含みません。 等号によって構造体とクラスが新しい変数にコピーされる方法にも違いがあります。

今のところ、デバイスからの応答をローカル変数に格納します。

```
#include <ZaberAscii.h>

ZaberShield shield(ZABERSHIELD_ADDRESS_AA);
ZaberAscii za(shield);

void setup() {
    /* Initialize baudrate to 115200, typical for Zaber ASCII devices */
    shield.begin(115200);
}

void loop() {
    /* Issue a get position command to the device */
    za.send(1, "get pos");

    /* Create a variable to store the device response inside */     // <-- This line is new.
    ZaberAscii::reply posResponse = za.receive();                   // <-- This line is new.
}
```

---

レスポンスからのデータは、発行された "get pos"コマンドからのデバイスの位置が含まれているために必要です。 これは、応答構造体の long int responseData 変数にあります。 したがって、変数に if 文を使用して、それが 0 かそれ以外の値かを確認します。

---

```
#include <ZaberAscii.h>

ZaberShield shield(ZABERSHIELD_ADDRESS_AA);
ZaberAscii za(shield);

void setup() {
    /* Initialize baudrate to 115200, typical for Zaber ASCII devices */
    shield.begin(115200);
}

void loop() {
    /* Issue a get position command to the device */
    za.send(1, "get pos");

    /* Create a variable to store the device response inside */
    ZaberAscii::reply posResponse = za.receive();

    /* Check if the device is currently at the home position */    // <-- This line is new.
    if (posResponse.responseData == 0)                             // <-- This line is new.
    {                                                              // <-- This line is new.
                                                                   // <-- This line is new.
    }                                                              // <-- This line is new.
    else                                                           // <-- This line is new.
    {                                                              // <-- This line is new.
                                                                   // <-- This line is new.
    }                                                              // <-- This line is new.
}
```

最後に、デバイスがホームポジションにある場合、LED ライトを消すコマンドを発行できます。 デバイスがホームポジション(else{ }の内側)にない場合は、引き続きデバイスを監視し、LED を点灯させます。

```
#include <ZaberAscii.h>

ZaberShield shield(ZABERSHIELD_ADDRESS_AA);
ZaberAscii za(shield);

void setup() {
    /* Initialize baudrate to 115200, typical for Zaber ASCII devices */
    shield.begin(115200);
}

void loop() {
    /* Issue a get position command to the device */
    za.send(1, "get pos");

    /* Create a variable to store the device response inside */
    ZaberAscii::reply posResponse = za.receive();

    if (posResponse.responseData == 0)
    {
        /* Set the device's LED lights to OFF if it is at home */     // <-- This line is new.
        za.send(1, "set system.led.enable", 0);                      // <-- This line is new.
    }
    else
    {
        /* Set the device's LED lights to ON */                       // <-- This line is new.
        za.send(1, "set system.led.enable", 1);                      // <-- This line is new.
    }

    /* Always read the reply, even if not using it. */                // <-- This line is new.
    /* Since each of the if/else blocks above only sends one */       // <-- This line is new.
    /* command, we can read either reply outside the blocks, here. */ // <-- This line is new.
    za.receive();                                                     // <-- This line is new.
}
```

これであなたは Arduino にこのプログラムをアップロードすることができます、そしてあなたはホームポジションにあるどんな装置でもその電源 LED ライトと通信 LED ライトが消えることに気付くでしょう。 (ノブの 1 つを使用して)デバイスをホームポジションから遠ざけると、LED ライトが再び点灯して監視できるようになります。

---

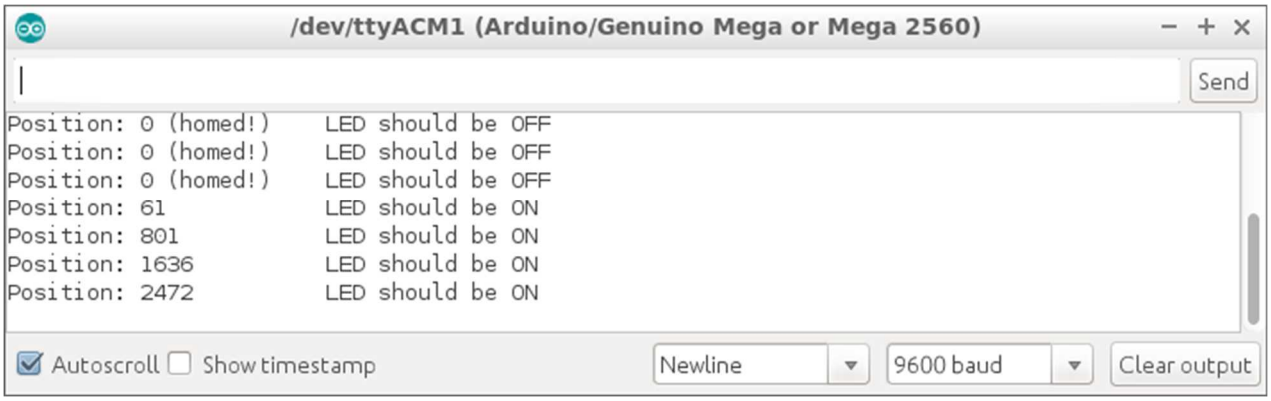
## シリアルモニタを使ってコードをデバッグ

X シリーズシールドを使用して Arduino に接続する利点の 1 つは、X シリーズシールドが Arduino のシリアルポートを使い切らないことです。

これにより、Zaber デバイスとの通信と同時にデバッグ情報をシリアルモニターに印刷できます。
前述の例の上に追加:

```
#include <ZaberAscii.h>

ZaberShield shield(ZABERSHIELD_ADDRESS_AA);
ZaberAscii za(shield);

void setup() {
    /* Initialize baudrate to 115200, typical for Zaber ASCII devices */
    shield.begin(115200);

    /* Initialize Arduino's serial port for debugging */            // <-- This line is new.
    Serial.begin(9600);                                             // <-- This line is new.
}
void loop() {
    /* Issue a get position command to the device */
    za.send(1, "get pos");

    /* Create a variable to store the device response inside */
    ZaberAscii::reply posResponse = za.receive();

    if (posResponse.responseData == 0)
    {
        /* Set the device's LED lights to OFF if it is at home */
        za.send(1, "set system.led.enable", 0);

        /* Send a debug message to the Serial Monitor */            // <-- This line is new.
        Serial.print("Position: 0 (homed!)¥t");                     // <-- This line is new.
        Serial.println("LED should be OFF");                        // <-- This line is new.
    }
    else
    {
        /* Set the device's LED lights to ON */
        za.send(1, "set system.led.enable", 1);

        /* Send a debug message to the Serial Monitor */            // <-- This line is new.
        Serial.print("Position: ");                                 // <-- This line is new.
        Serial.print(posResponse.responseData);                     // <-- This line is new.
        Serial.println("¥t¥tLED should be ON");                     // <-- This line is new.
    }

    /* Always read the reply, even if not using it. */
    /* Since each of the if/else blocks above only sends one */
    /* command, we can read either reply outside the blocks, here. */
    za.receive();

    /* Add a delay to limit polling to 2 times per second */        // <-- This line is new.
    delay(500);                                                     // <-- This line is new.
}
```

それでは、プログラムを Arduino にアップロードし、Arduino IDE でシリアルモニターを開きましょう。これを行うには、[Tool] - > [Serial Monitor]をクリックするか、Ctrl + Shift + M のショートカットを使用します。コードと一致するようにボーレートが 9600 に設定されていることを確認してください。シリアルポートがまだ IDE で正しく設定されていると仮定すると、以下のような印刷が表示されるはずです。毎秒約 2 回更新されます。




シリアルプロッタ([ツール] - > [シリアルプロッタ]、または[Ctrl] + [Shift] + [L])を使ってデバイスの位置をグラフィカルに追跡することもできます。

---

## 返信の解釈

デバイスの返信を解釈する方法に関するいくつかの追加のコメント。 前述のように、応答データ構造には以下のフィールドが含まれています。

- String fullResponse : 未解析のレスポンス文字列全体。
- bool isReply : 応答が返答であれば真と評価される
- int deviceNumber : コマンド応答が関連していることをデバイス番号に評価します。
- int axisNumber : コマンド応答が関連する軸番号に評価されます。
- bool isRejected : コマンドがデバイスによって拒否された場合は true を返します。
- bool isBusy : デバイスがビジー状態でコマンドを受信できなかった場合は true を返します。
- bool hasWarning : コマンドがデバイスから警告を表示した場合は true を返します。
- String warningFlags : 警告が発行された場合は警告フラグを含む文字列
- String responseDataString : 文字列変数でレスポンスに添付されたデータ
- long int responseData : 整数変数内の応答に添付されたデータ。

コマンドに対する正しい応答を確認していることを確かめたい場合は、deviceNumber がメッセージの取得先と予想されるデバイスと一致していることを確認する必要があります。それから isReply は true です。 isReply が false の場合は、アラートまたは情報メッセージがあることを意味します。これらのメッセージタイプを処理する場合は、fullResponse の最初の文字を確認する必要があります。アラートは '!' で始まります 情報メッセージは '#'で始まります。

返信にデータを使用する前に、コマンドが正常に実行されたことを確認する必要があります。isRejected をチェックしてデバイスがコマンドを拒否していないことを確認し、isBusy をチェックしてデバイスが現在移動中かどうかを確認します。ZaberAscii クラスの isIdle または pollUntilIdle メソッドを使用して、このデータを自分で解釈しなくても、移動/アイドル状態を確認できます。

データ値を返すほとんどのコマンドでは、応答の Data フィールドに値が数値形式で表示されます。ただし、responseDataString フィールドから自分でデータを解析する必要がある場合もあります。返されるデータに小数点以下の桁数、複数の値、32 ビット符号付き整数に収まらないほど大きい数、または数値以外の場合 データ。

hasWarning および warningFlags フィールドは、デバイスが警告状態を報告しているかどうかを示します。 あなたが気にすべき条件を示しているかどうかは、どちらのフラグが存在するか、そしてあなたの使用状況によって異なります。解釈については、Zaber ASCII プロトコルマニュアルの警告フラグのセクションを確認してください。また、複数の警告が一度にアクティブになることがありますが、返信には通常最も緊急の警告のみが含まれます。「warnings」コマンドを使用してそれらすべてを取得できます。

返信条件の解釈の基本的な例や、返信を待っている間にプログラムがブロックされないようにするもう 1 つの方法については、「手動移動の追跡」サンプルプログラムを参照してください。

---

### より多くのシールドとセンサーを追加する場合

新しいハードウェアを接続する前に、すべての電源を切ることをお勧めします。 セットアップに追加するときは、次の点を考慮してください。

1. **I2C アドレス：他の shield のいずれかが Arduino と通信するために I2C を使用している場合....**
   - 以下の 4 つのオプションから、X シリーズシールドに固有のアドレスを選択してください：
     - I2C address AA: 0x90
     - I2C address AB: 0x92
     - I2C address BA: 0x98
     - I2C address BB: 0x9A
   - アドレスを変更した場合は、有効にするためにボードの電源を入れ直す必要があります。 単純なリセットは機能しません。
   - ZaberShield クラスの初期化で指定されているコード内のアドレス（ZABERSHIELD_ADDRESS_xx）が、ジャンパー設定と一致していることを確認してください.
2. **電源**：コードに満足したら、Arduino をコンピューターから取り外して X シリーズ Zaber デバイスから電源を入れることができます。
   - この時点で、電源ジャンパを「X シリーズ電源」の位置に切り替えることができます。.
   - バレルジャックの電源入力または Vin ピンを介して Arduino に電源を供給したい場合は、ジャンパを「Vin / USB」の位置にしておく必要があります。
3. **堅牢な I / O:** 接続している余分な信号を少し保護したい場合は、この付録を読んでください。

## 私のプログラムがうまく動作しない！

**プログラムは Arduino ボードにアップロードされない。**

- [Tool] - > [Board]、および[Tool] - > [Port]で、ポートとボードの設定を確認して下さい.

- Zaber X シリーズシールドを使用している場合は、シールドに電源が入っていることを確認してください。 右下隅にある ON の LED が点灯します。 電源ジャンパが配線と一致していることを確認してください。
- Zaber 製以外の RS-232 シールドを使用している場合は、シールドがオフになっていることを確認してください。 RS-232 シールドにオン/オフスイッチがない場合は、Arduino にアップロードするときに外してください。 あなたのプログラムがアップロードされた後にそれを再びオンにすることを忘れないでください。

**デバイスが私のコマンドを認識していないか、ランダムに移動/反応しているようです。**

- 正しいプロトコルとボーレートを使用していることを確認してください。 デフォルトでは、X シリーズデバイスは ASCII、115200 ボーレート、T シリーズおよび A シリーズデバイスはバイナリ 9600 ボーレートです（A-MCB2 を除く）。 Zaber コンソールでプロトコルとボーレートを確認し、オプションタブを使用して 2 つのプロトコルを切り替えることができます。 Zaber コンソールの[オプション]タブでは、デバイスのプロトコルを簡単に変更できます。また、正しいコマンドを送信していることを確認してください。 ASCII プロトコルでは、コマンドは ″get pos″のように文字列として識別されます。ASCII プロトコルのコマンドの全リストはここにあります。

**期待している返答が得られない、または返答が多すぎる**

- 各機器から返されたすべての応答を読んでいることを確認してください。 ほとんどの場合、Zaber デバイスは受信したコマンドごとに 1 つの応答を送信します。基本的な使用には、このように処理すれば通常は十分です。
- 特定の条件下では、デバイスはコマンドに応答しないメッセージを自発的に送信できます。 ASCII プロトコルでは、条件が満たされたときに警告メッセージを送信するようにデバイスを設定することができます。また、いくつかのコマンドはそれらの応答に複数の情報メッセージを追加します。 Zaber ライブラリはこれらの自発的なメッセージを自動的に処理しないので、それらを生成する可能性のあるデバイスを使用している場合は、返信を読むときにそれらを確認し、最後のコマンドに対する期待される返信としてカウントしないことを認識してください。

**動くたびに、デバイスは高音のビープ音を鳴らしています**

お使いのデバイスが正しく設定されていることを確認するには、contact @ zaber.com で Zaber サポートに連絡してください。

**これらの修正は私にはうまくいきませんでしたし、私のプログラムはまだ実行できません。**

Arduino のボードタイプ、Zaber デバイスモデル、Zaber または RS-232 シールドタイプ、および実行を拒否する Arduino コードの詳細については、contact @ zaber.com で Zaber サポートにお問い合わせください。

# 追加のプログラム例

その他のプログラム例は Arduino IDE を通してアクセス可能です。 このライブラリに含まれるすべてのプログラム例を見つけるには、メニューシーケンス File - >Example - > Zaber ASCII を使用してください。 各例には、その動作を説明するコメントが含まれています。

# 付録 A：堅牢な I / O インタフェースの使用

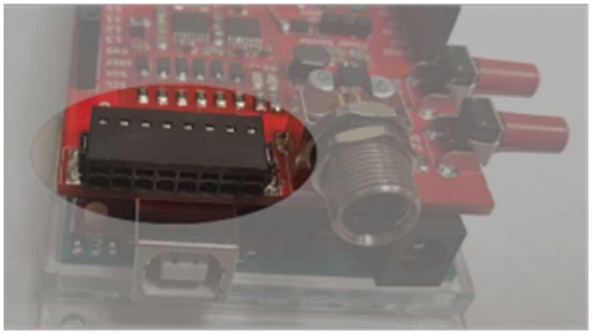
Arduino の入力と出力の一部を高耐久化し、保護された信号を黒い端子台に表示することを選択しました。

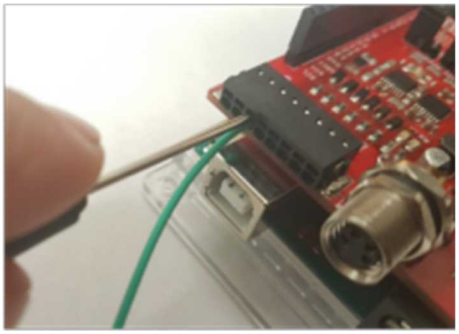
彼らは防弾ほどの効力はありませんが、あなたの Arduino の供給電圧の外側に数ボルトが掛かっても、Arduino が損傷受けることはありません。 可能な限り保護された端子台の入力/出力を使用することをお勧めしますが、Arduino の動作を初期化するためにいつでもピンヘッダに直接接続することができます。

堅牢化されたインターフェースにより、以下の機能にアクセスすることができます。

- 2 つのアナログ入力†(A0 と A1)
- 3 つの PWM 出力(デジタルピン 3、5、および 6)
- 4 つのデジタル入力または出力(デジタルピン 3～6)
- +5V。150mA に制限されているため、さらに電流を引き込むと無効になります。

†保護されたアナログ入力が読み取れる最大アナログ値は、1024 のうち 1020 です。





このコネクタに接続するには、単純に接続ホールに所要のワイヤーを差し込んでください。 バネ付クランプで固定されます。

ワイヤを取り外すには、X シリーズシールドに付属の小さなマイナスドライバを使ってワイヤの上の狭い穴に押し込みます。

詳細については、X-AS01 ハードウェアの概要をご覧ください。

# 付録 B:Zaber の X シリーズシールドなしで使用開始

弊社の推奨 X シリーズシールドがない場合

1. Arduino の上に RS-232 シールドを取り付け、付属の USB ケーブルを使用して Arduino ボードをコンピュータに差し込みます。
2. Zaber デバイスを RS-232 シールドに接続するには、最初にヌル M / M アダプタを使用して、メス DSUB9 をオス DSUB9 に変換する必要があります。
3. T シリーズまたは A シリーズの Zaber デバイスを使用している場合は、T-DSUB9 アダプタも必要です。 X シリーズデバイスを使用している場合は、X-SDC ケーブルが必要になります。Arduino にアップロードするときは、RS-232 コンバータがオフになっていることを確認してください(スイッチは上部にあります)。T シリーズの Zaber デバイスは ASCII プロトコルをサポートしていません。 Arduino で T シリーズデバイスを制御したい場合は、Arduino 用の Zaber Binary ライブラリを使用する必要があります
4. コード内で ZaberAscii クラスの代わりに Serial クラスを使用します。 すべて同じメソッドを持っていますが、ポートアドレスをそのコンストラクタに渡す必要はありません。

# ドキュメンテーション

Zaber の X-Series Shield の回路図と短いハードウェアリファレンスはここにあります。.

Arduino 用の Zaber のすべてのライブラリのドキュメントは、ZaberShield.h および ZaberAscii.h ファイルにあります。 Windows PC では、これらのファイルの一般的なインストール場所は Documents ¥ Arduino ¥ libraries ¥ ZaberAscii です。

テクノロジーリンク株式会社
TECHNOLOGYLiNK,LTD.
〒171-0022 東京都豊島区南池袋 3-18-35
OK ビル 2 階
Tel: 03-5924-6750 Fax:03-5924-6751
E-mail: sales@technology-l.com
URL: http://www.technology-link.jp