**LED点灯制御**

**（C＃アプリ）**

**研修テキスト**

|  |  |
| --- | --- |
| 作成者 | デニスケビンデラクルーズ |
| 作成日 | 2020/10/28 |
| 最終更新日 | 2020/10/30 |
| 文書番号 |  |

パーソルテクノロジースタッフ株式会社

**目次**

[1. 目的 3](#_Toc54945869)

[1-1. 研修背景・目的 3](#_Toc54945870)

[2. 環境 4](#_Toc54945871)

[2-1. ソフト環境 4](#_Toc54945872)

[Visual Studio 2019 4](#_Toc54945873)

[2-2. ハードウェア環境 5](#_Toc54945874)

[3. 回廊設計 6](#_Toc54945875)

[3-2. ブレットボード 7](#_Toc54945876)

[4. 実装 8](#_Toc54945877)

[4-1. Arduino側 8](#_Toc54945878)

[4-2. Visual Studio側 13](#_Toc54945879)

# 目的

## 研修背景・目的

この研修テキストはC言語→C#への移行が円滑に学習できるように目標としています。 制御ではマイコンベースのC言語から使っているので、今回、C＃アプリから簡単なストリング通知でマイコンであるのLEDを点灯します。シリアルポートを介してデータを送信し、Arduinoで受信するC＃アプリケーションを作成する目標です。

表 1 CとC#の差分表では、CとC#の単純な差分が記載します。。プログラミングの世界では、C#がアプリケーション阻に適切です。一方、Cが組み込み制御に適切です。

表 1 CとC#の差分表

|  |  |
| --- | --- |
| C | C# |
| 手続き型プログラミング | オブジェクト指向プログラミング |
| 最低レベルの抽象化 | 高レベルの抽象化 |
| 手動メモリ管理 | ガベージコレクション |
| 組み込みデバイスおよびシステムレベルのコードに最適 | シンプルなWeb、モバイル、デスクトップアプリケーションに最適 |

C#と言えば、Windows環境と思われがちですが、.Net Coreをインストールすれば、Linux環境でもC#で開発したアプリケーションが動作します。また、Raspberry Pi（ラズベリーパイ）でも、C#を使ってIoT開発を行うことができます。この研修では、C#によるWindows環境アプリケーション層にIoTプログラミングで使用します。図 1 研修階層では、この研修の単純なIoT階層となり、トップダウンで作成します。

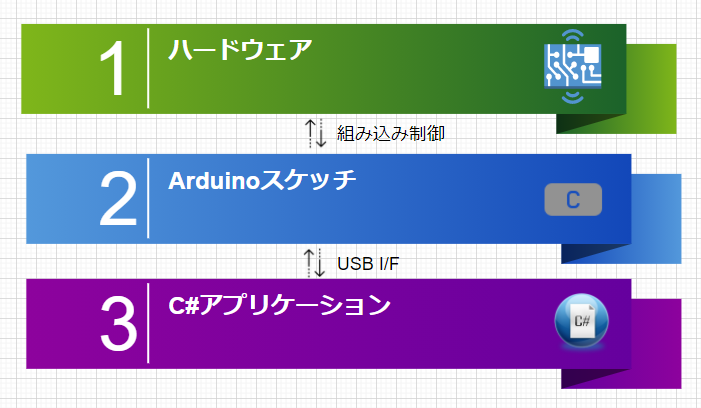


図 1 研修階層

# 環境

## 2-1. ソフト環境

Windowsパソコンを利用します。パソコンでは、表 1 ソフト一覧のソフトをダウンロードしてインストールしてください。

表 2 ソフト一覧

|  |  |  |
| --- | --- | --- |
| **ソフト名** | **リンク** | **目的** |
| Arduino IDE | <https://www.arduino.cc/en/main/software> | スケッチをArduinoにロードするため |
| Visual Studio 2019 ※コミュニティで良いです。 | <https://visualstudio.microsoft.com/ja/downloads/> | C#アプリを作成するため **注意:** Visual Studio インストーラーで、 [.NET デスクトップの開発] ワークロードを選択してください。選択しなかったら、[Windows フォーム アプリ (.NET Framework)] テンプレートが表示しない。 |

## 2-2. ハードウェア環境

表 2　ハードウェア覧に記載では、この研修の必要なハードウェアです。研修前に準備してください。(図 1　ハードウェア写真でも、実際な部品になります。)

表 3　ハードウェア覧

|  |  |  |
| --- | --- | --- |
| **部品種類** | **数** | **備考** |
| Arduino Uno | 1 |  |
| 100Ω抵抗 | 3 |  |
| LED | 3 | 1 – Red, 1 – Yellow, 1-Green |
| 銅線 | 4以上 |  |
| ブレットボード | 1 |  |
| シリアルI/F(Firewire->USB) | 1 | Arduinoがパソコンに接続するUSBで良いです。 |

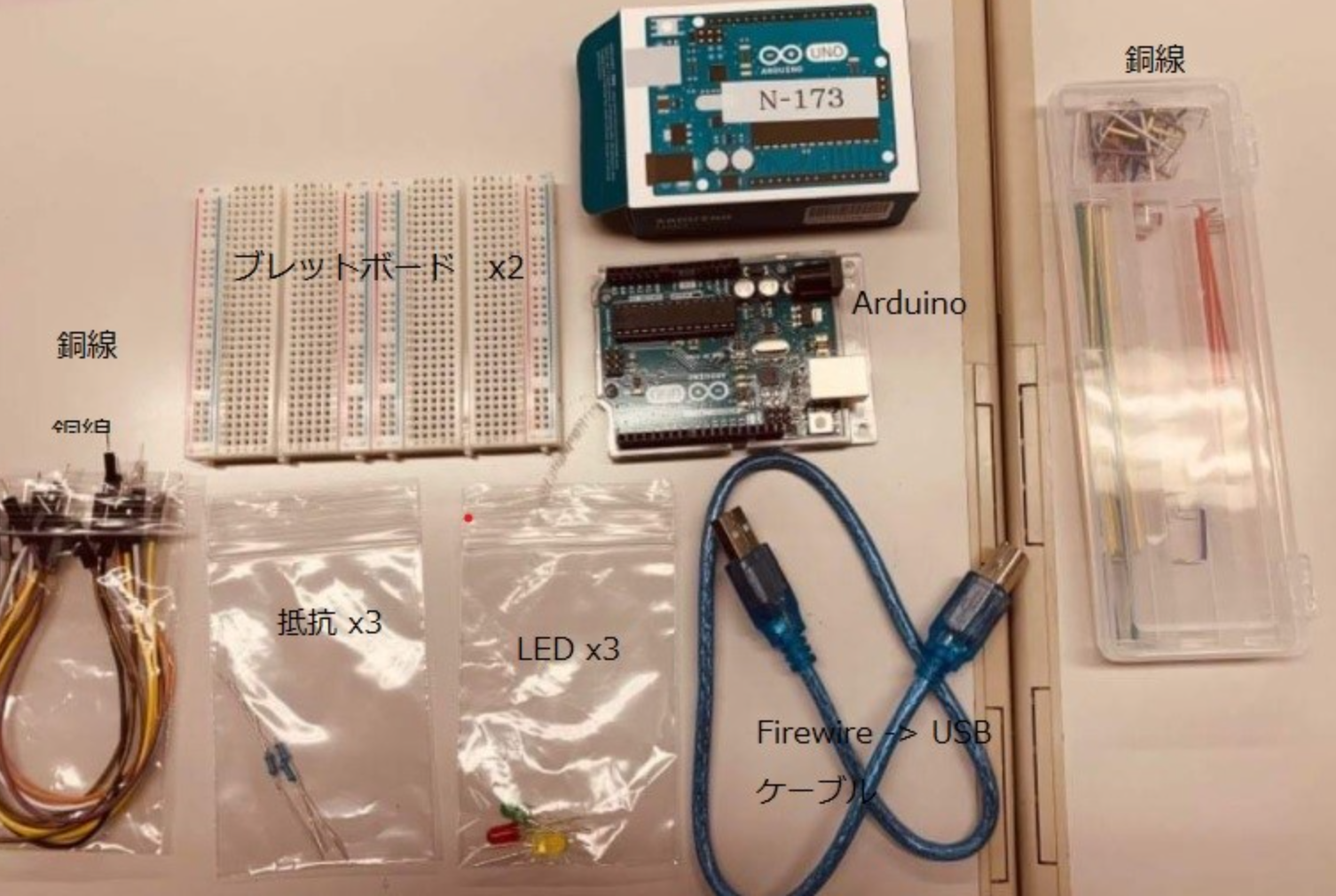


図 2　ハードウェア写真

# 回廊設計

3-1. 回路図

図 2　3つLEDの回路図では、3つの制御するLEDになります。下記のようにブレットボードに装着してください。基本的に、Arduinoで制御するピンは表 3 ピン装着になります。

図 3　3つLEDの回路図

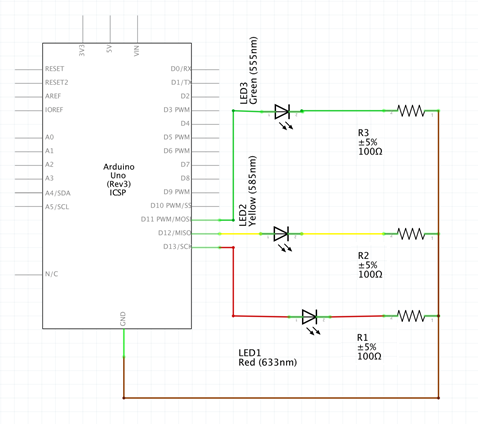


表 4 ピン装着

|  |  |
| --- | --- |
| **LED色** | **Arduinoピン** |
| RED | 13 |
| YELLOW | 12 |
| GREEN | 11 |

## 3-2. ブレットボード

部品はブレットボードに装着したら、図 3 ブレットボードの表現になります。実現したものについて、図 4 ブレットボード写真を参照ください。

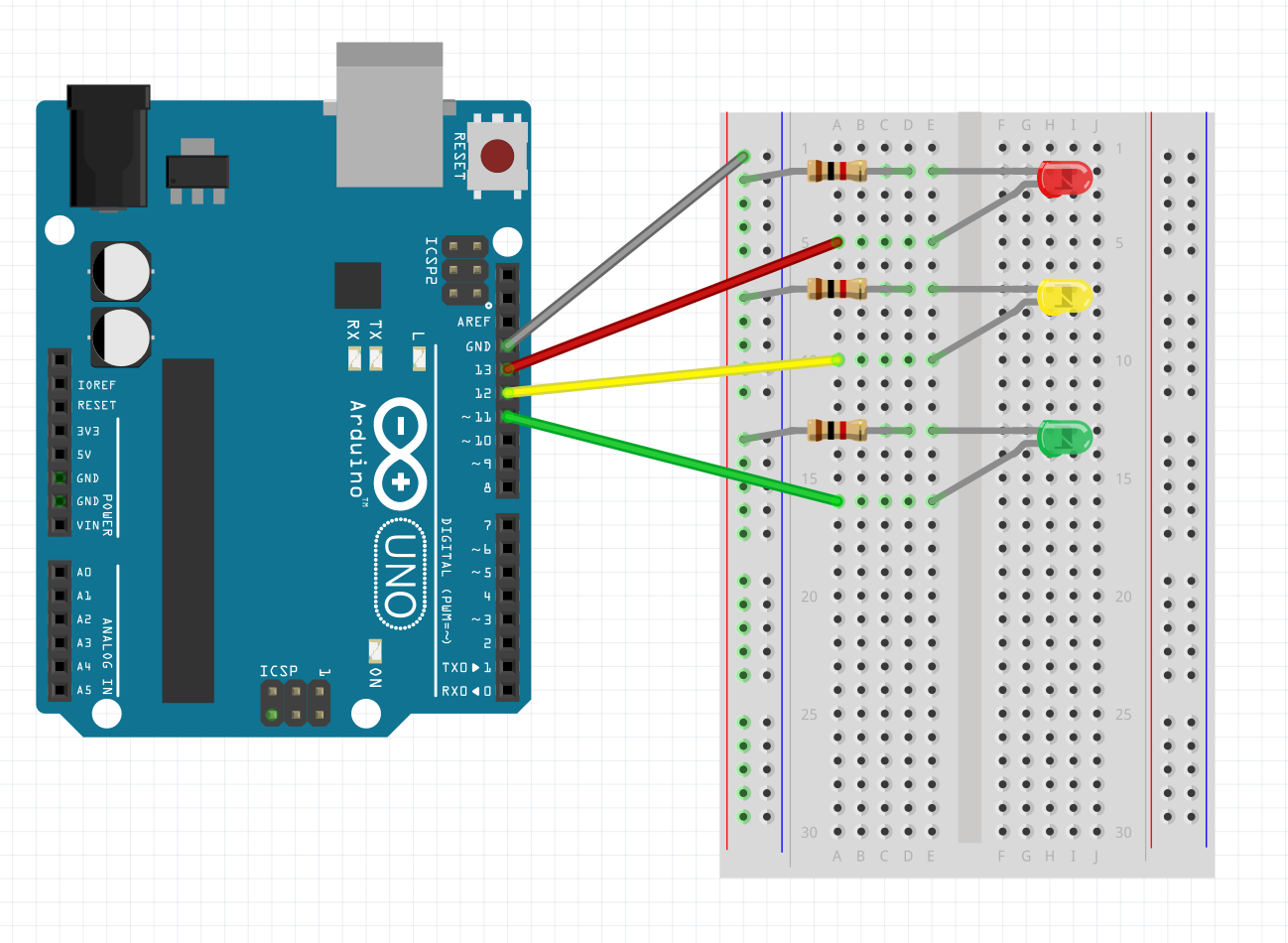


図 4 ブレットボードの表現

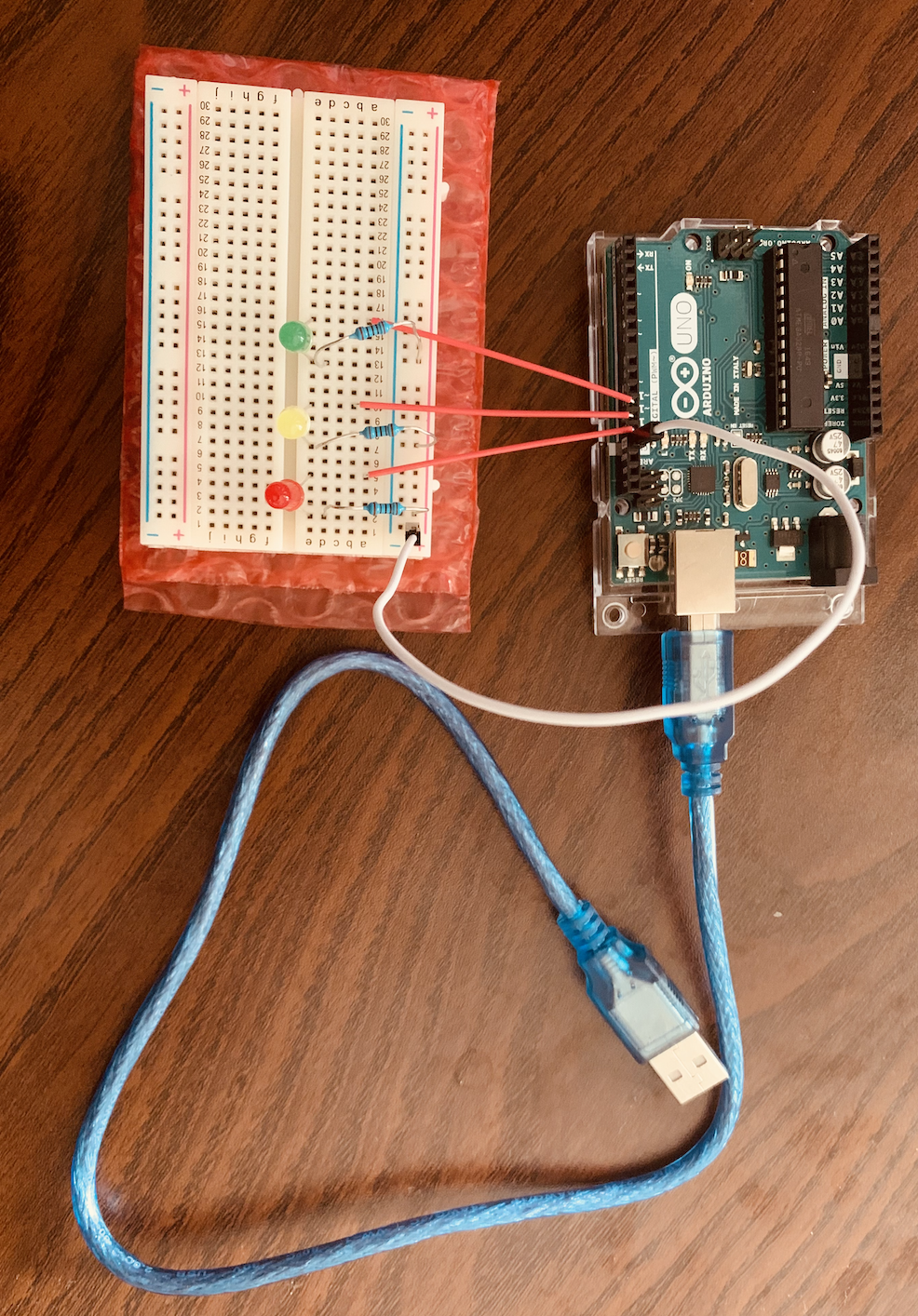


図 5 ブレットボード写真

# 実装

## 4-1. Arduino側

ArduinoはUSBでパソコンに装着してArduino IDEを開いてください。

Arduino IDEを開いたら、図 5 Arduino IDE起動時が表示します。

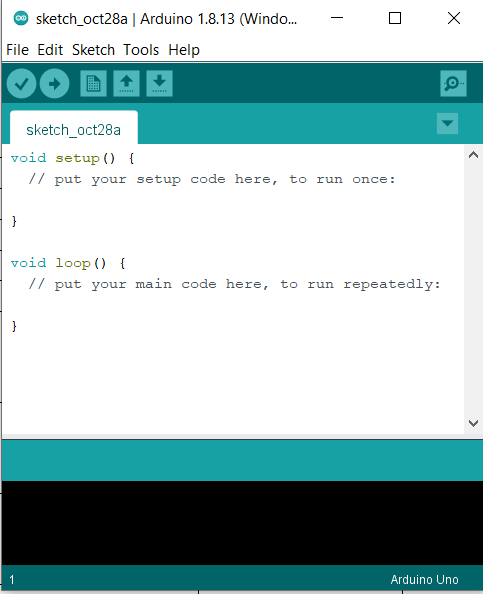


図 6 Arduino IDE起動時

Arduinoを接続するため、「Tools」->「Port」に「COM＊」を選択します。（図 6 Arduinoポート確認を参照）

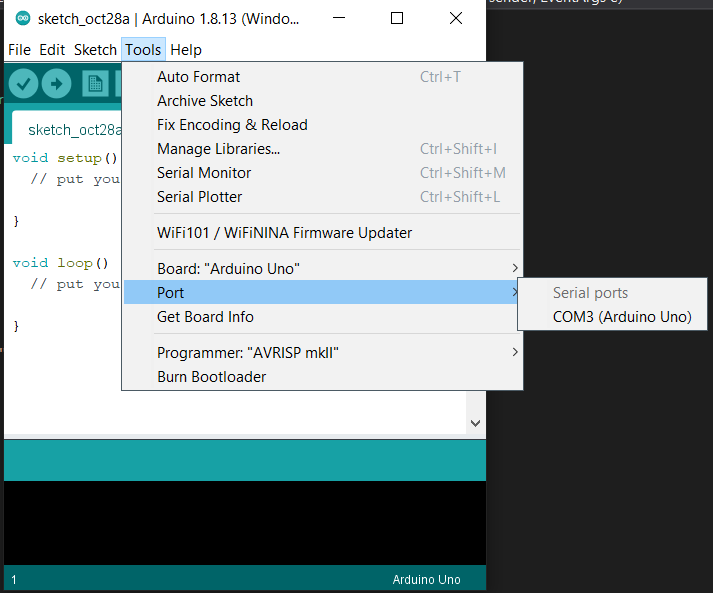


図 7 Arduinoポート確認

Arduino IDEでは、下記部分を実装します。

1. 変数の定義
2. Setup関数
3. Loop関数

表 4 定義表のように変数を定義します。

表 5 定義表

|  |  |  |  |
| --- | --- | --- | --- |
| **定義名** | **定義種類** | **定義値** | **備考** |
| RED\_PIN | const int | 13 | 表 3 ピン装着により |
| YELLOW\_PIN | const int | 12 | 表 3 ピン装着により |
| GREEN\_PIN | const int | 11 | 表 3 ピン装着により |
| BAUD\_RATE | const int | 9600 | I/F通信のbaudRate |
| RED | const int | 0b000001 | Binaryで立ち状態のであればREDのLEDは点灯 |
| YELLOW | const int | 0b000010 | Binaryで立ち状態のであればYELLOWのLEDは点灯 |
| GREEN | const int | 0b000100 | Binaryで立ち状態のであればGREENのLEDは点灯 |
| incomingOption | char | - | SerialI/Fからの受け取った文字 |
| option | int | - | 通信データの変換した値 |
| range | int | - | LED点灯状態(0~7) |

図 5 Setup関数Arduinoスケッチのsetup関数です。基本的に装着したPINがOUTPUTに設定して、Serialのbaud rateは9600で初期します。



図 8 Setup関数

図 6 loop関数はArduinoスケッチのloop関数です。



図 9 loop関数

実装した後、ソースコードはArduinoにアプロードしてください。Arduino側のソースコードは下記のリンクもダウンロードできます。

<https://github.com/dktdc91/-/blob/main/LEDSketch.c>

ソースコードの動作を確認すると、「Serial Monitor」では、「1」を送信するとREDのLEDが点灯されます。(図 9 Serial Monitorを参照)

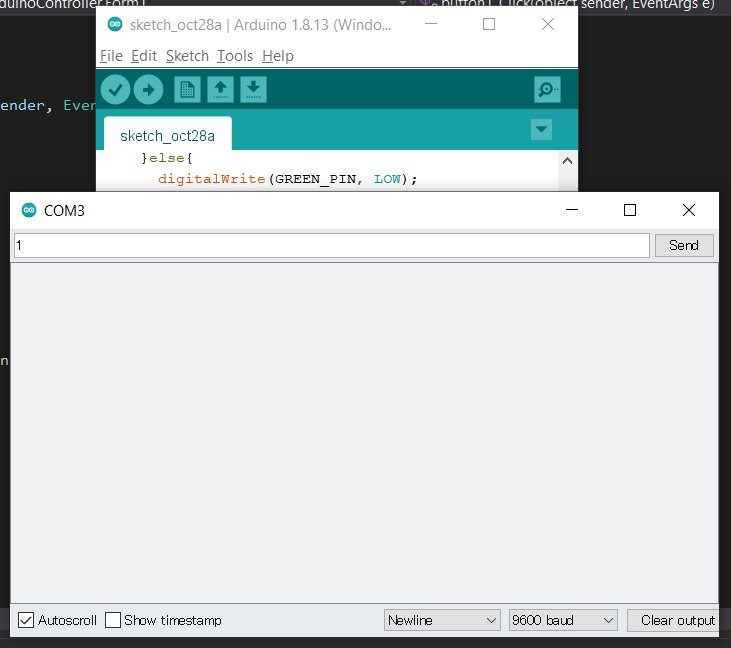


図 10 Serial Monitor

## 4-2. Visual Studio側

Visual Studio 2019を開いてください。

Visual Studio 2019を開いたら、図 10 Visual Studio 2019開始画面が表示します。「Create a new project」を選択してください。

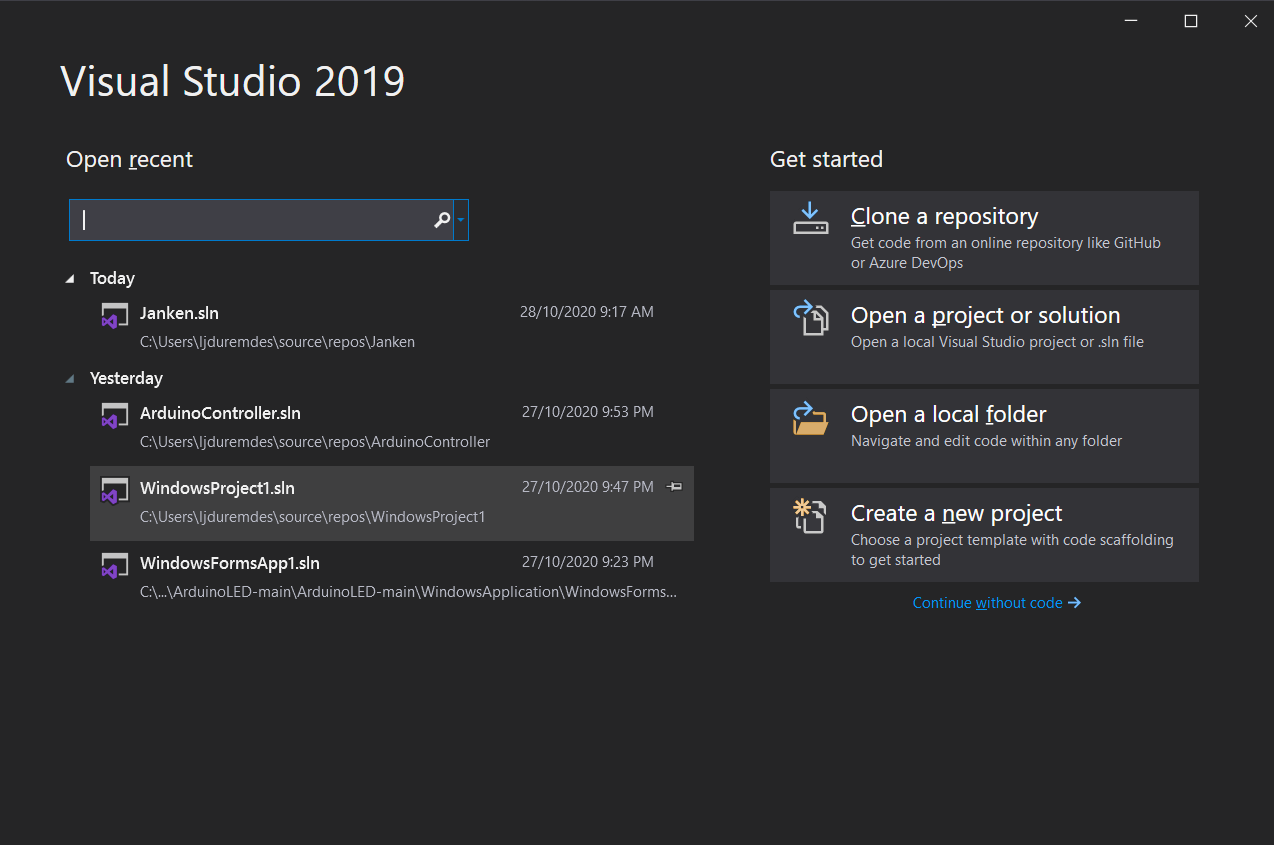


図 11 Visual Studio 2019開始画面

次の画面では、「Windows Forms App (.NET Framework)を選択してください。（図 11　新規プロジェクト作成画面を参照）

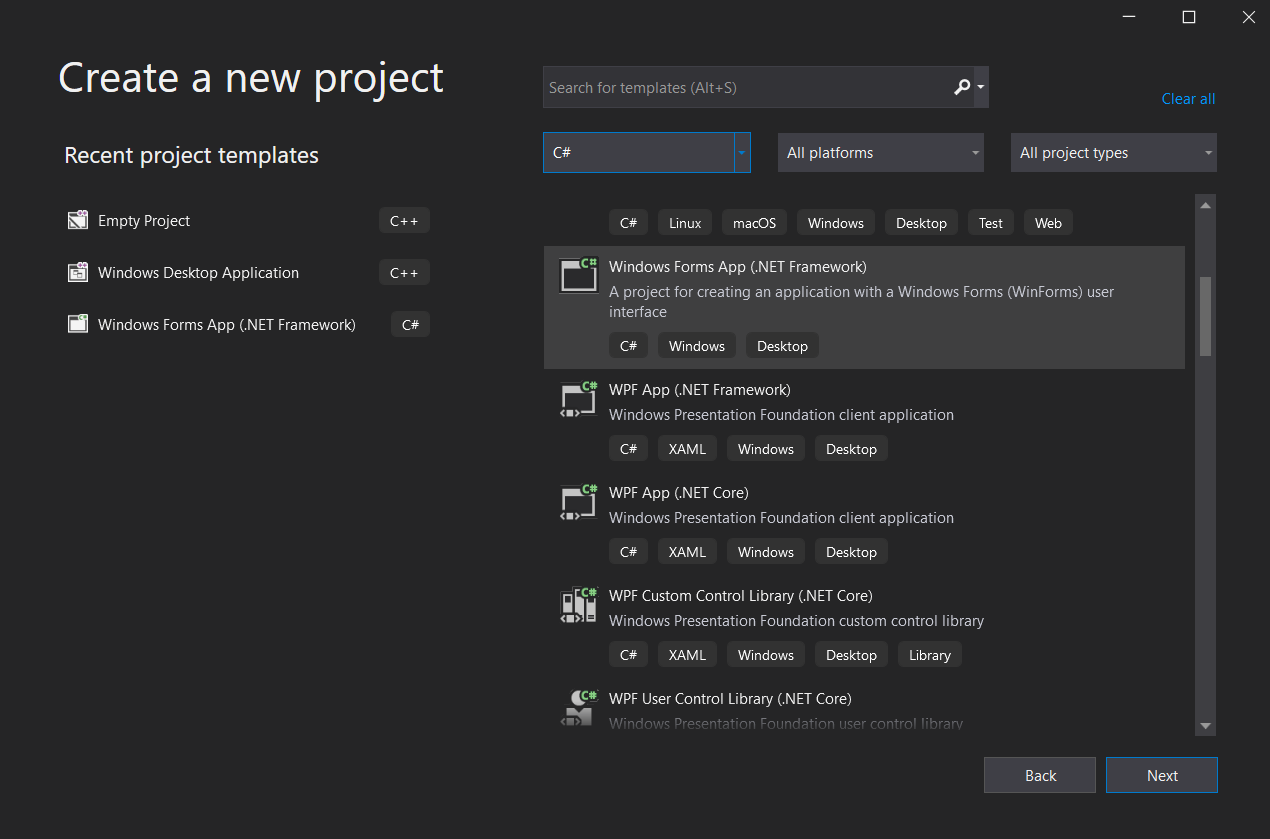


図 12　新規プロジェクト作成画面

次の画面では、任意なプロジェクト名、Solution名を入力して、「Create」をクリックしてください。(図 12　新たなプロジェクト構成画面を参照)

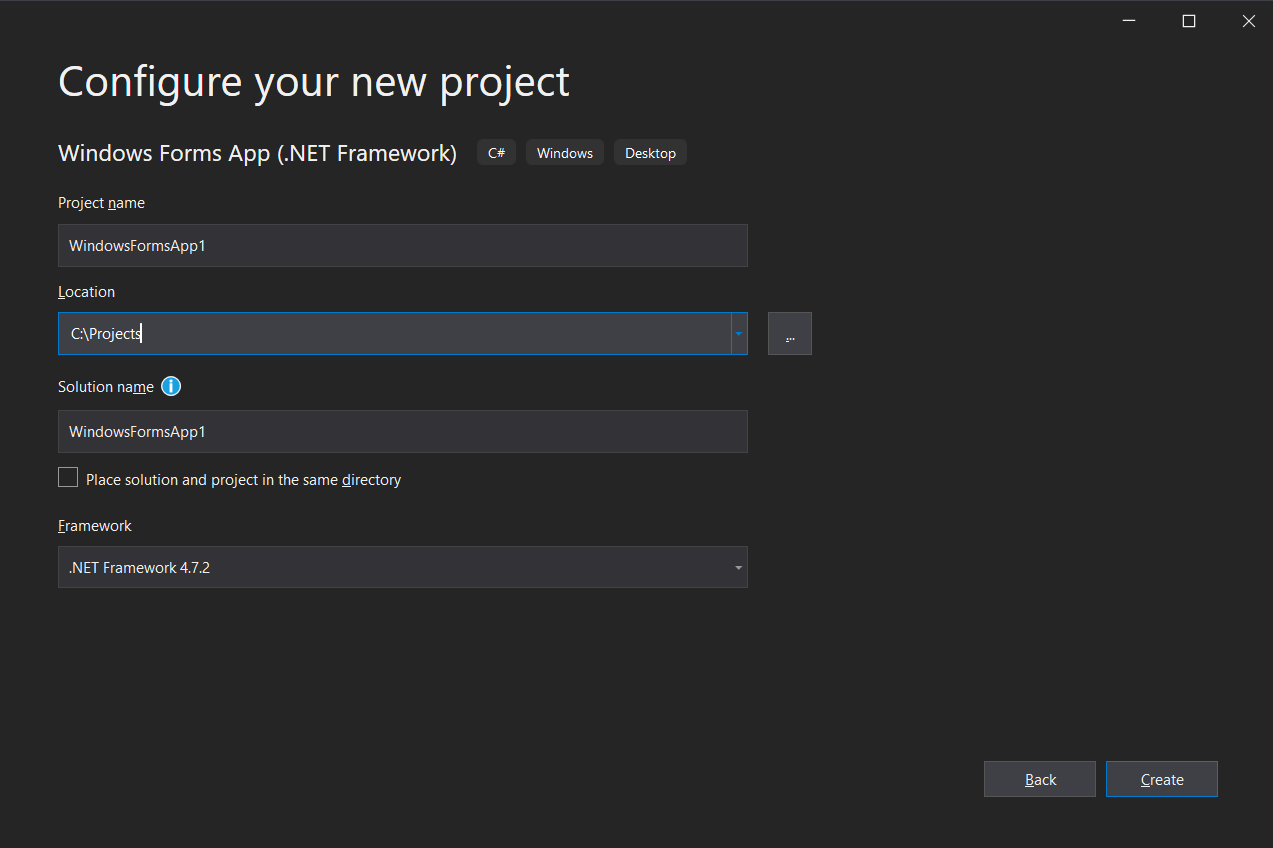


図 13　新たなプロジェクト構成画面

作成した後、下記の画面に移動します。

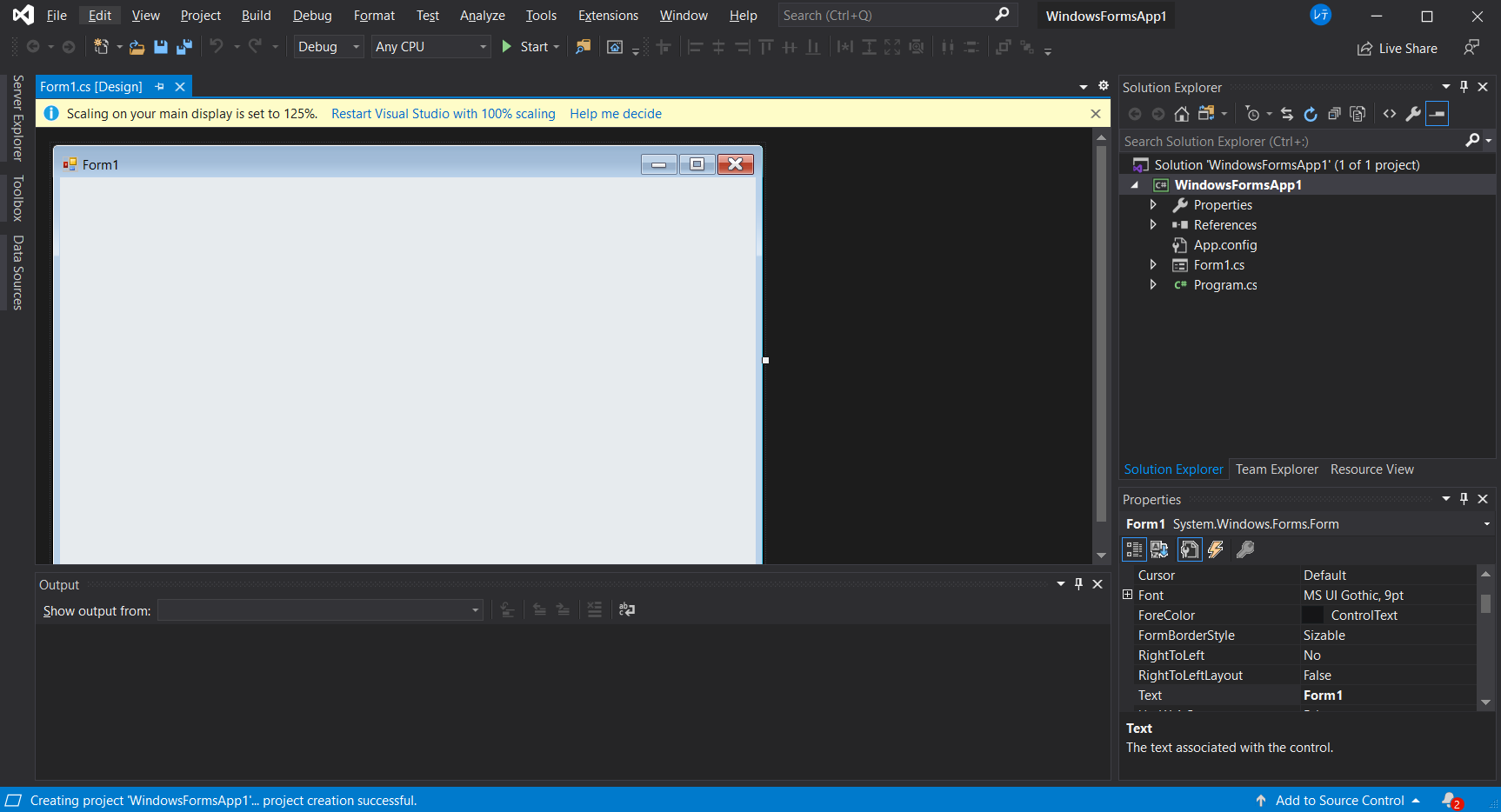


図 14　Designer View

ツールボックスでは、3つのチェックボックスと1つのボタンを追加します。

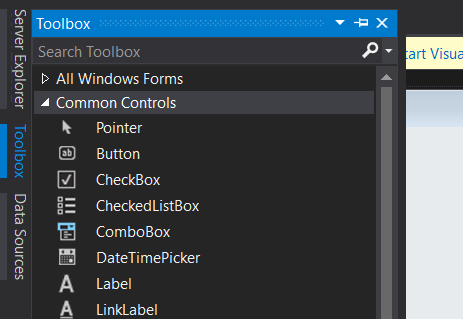


図 15 ツールボックス

追加した後、フォームが図 15 Designer Viewサンプルのようになります。

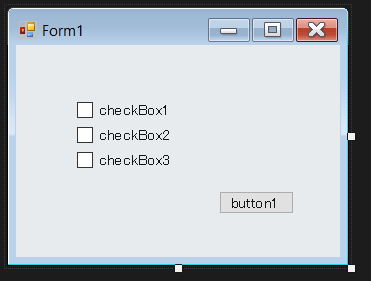


図 16 Designer Viewサンプル

チェックボックス・ボタンのメッセージを変更するため、追加したコンポネントをクリックして、プロパティ部分では「Text」(図 16　プロパティ変更図を参照)を変更してください。表 5 チェックボックス名称変更のように変更してください。

表 6 チェックボックス名称変更

|  |  |
| --- | --- |
| **前** | **後** |
| checkBox1 | Red |
| checkBox2 | Yellow |
| checkBox3 | Green |
| button1 | 送信 |

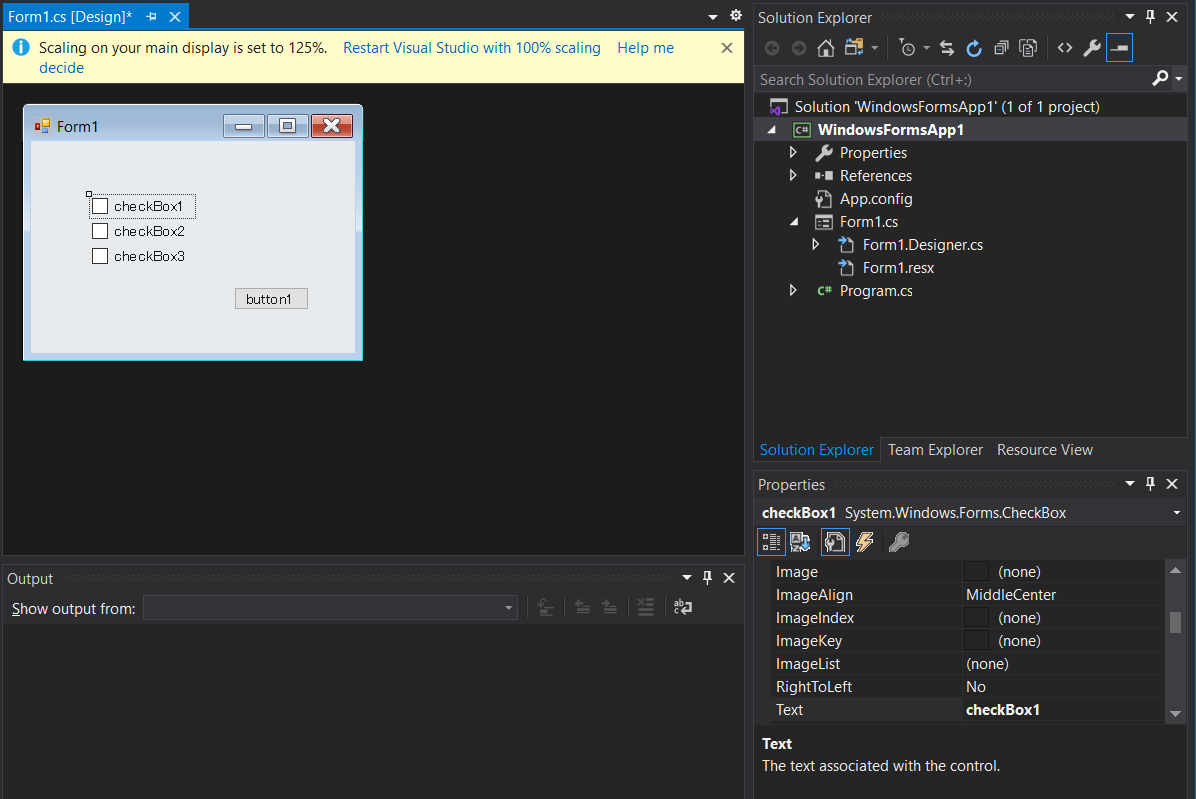


図 17　プロパティ変更図

変更が終わったら、図 17 変更後Designer Viewサンプルのようになります。

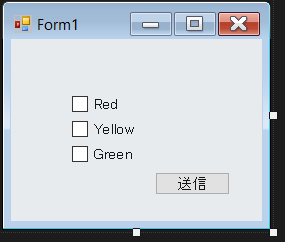


図 18 変更後Designer Viewサンプル

ソースコードでは、下記の変数を定義します。(ソースコードに遷移するため、「View」→ 「Code」それとも、F7を押す)

表 7 アプリの定義表

|  |  |  |  |
| --- | --- | --- | --- |
| **定義名** | **定義種類** | **定義値** | **備考** |
| BAUD\_RATE | const int | 9600 | I/F通信のbaudRate |
| RED | const int | 0b000001 | Binaryで立ち状態のであればREDのLEDは点灯 |
| YELLOW | const int | 0b000010 | Binaryで立ち状態のであればYELLOWのLEDは点灯 |
| GREEN | const int | 0b000100 | Binaryで立ち状態のであればGREENのLEDは点灯 |
| port | private SerialPort | - | シリアルポートオブジェクト using System.IO.Ports; |
| port\_name | private static string | "COM3" | Arduinoの接続したポートによって定義値を変更してください。 |
| led\_state | private int | 0b000000 | 通信するled状態値 |

下記のように実装されます。

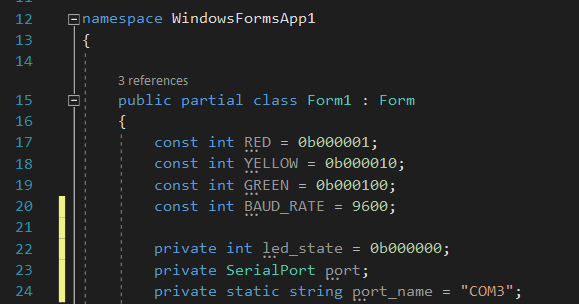


図 19 定義実装

初期化の時、通信のためportのオブジェクト作成するのは必要です。作成するため、下記のように実装します。

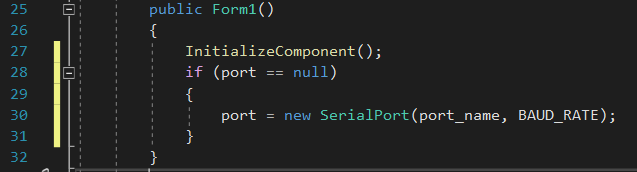


図 20 ポートの初期化

チェックボックスの状態により、LEDを点灯するため、チェックボックスの値が変わったら、「led\_state」を更新します。チェックボックスのアクションのため、「View」→ 「Designer」に遷移します。

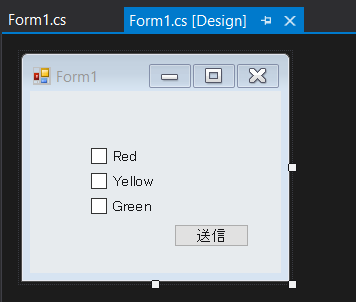


図 21 Designer View移動

Redのチェックボックスのコンポネントをダブルクリックしたら、「Code」Viewに遷移し、あらたな「checkBox1\_CheckedChanged」関数が作成します。

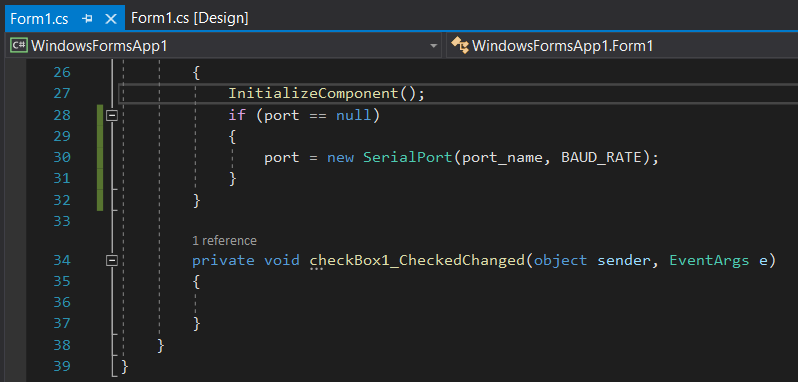


図 22 自動作成関数

この時、チェックボックス値によって、「led\_state」がLEDのビットを更新する。コードは下記になります。

led\_state ^= RED;

led\_state ^= YELLOW;

led\_state ^= GREEN;

ソースコードは図 22 全チェックボックス実装のようになります。注意: Visual Studioはチェックボックスのアクション機能が自動的に接続していますので、Designerビューのダブルクリックは必要です。

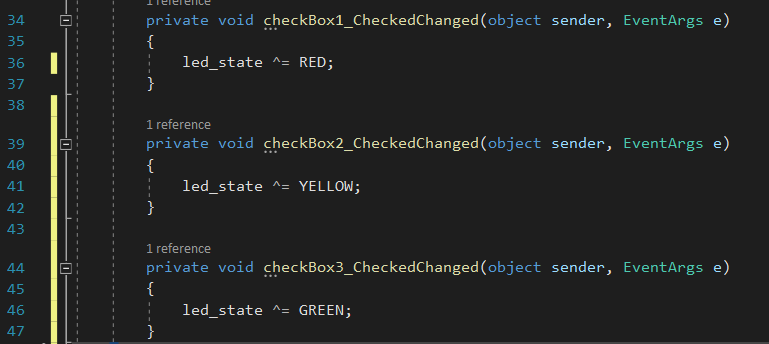


図 23 全チェックボックス実装

次は、「led\_state」がArduinoに送信します。このため、Designerビューでは「送信」ボタンをダブルクリックして、アクション関数をつくります。

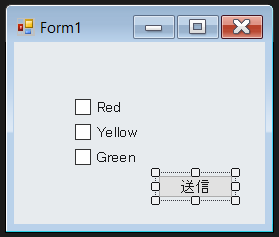


図 24 送信ボタン

作成した関数の中に、portを開いて、「led\_state」がstringで送ります。実装すると、図 24　ポート通信の実装のようになります。

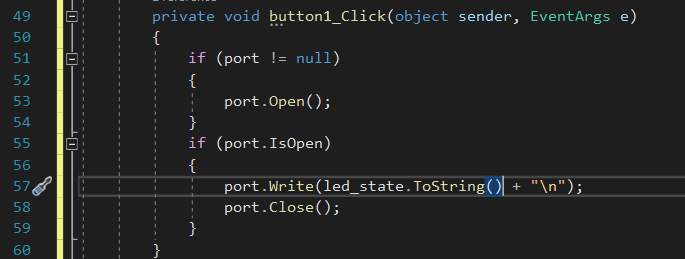


図 25　ポート通信の実装

作成したプログラムを実行するため、下記のボタンを押下してください。

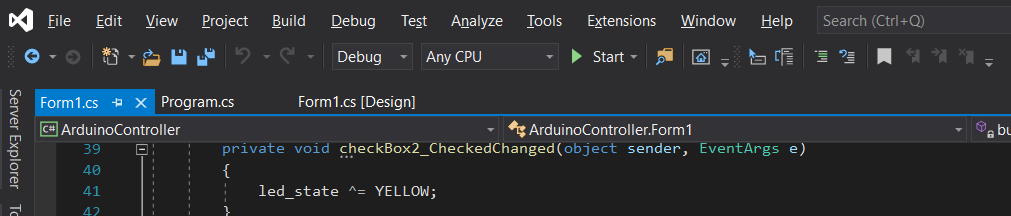


図 26 Debug実行

ビルドエラーがなかったら、Designer Viewのようにプログラムが表示するはずです。

注意: 送信ボタンが押す前に、Arduino IDEの「Serial Monitor」を閉じてください。