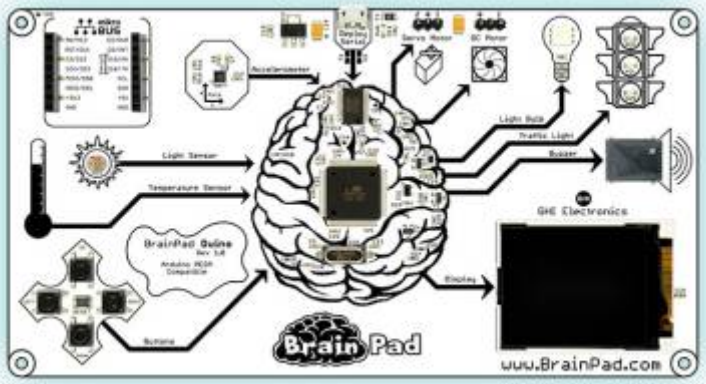
Les classes de la carte BrainPad v1

[Mise à jour le 18/1/2019]

La carte BrainPad v1



Sources : documentation sur le site de GHI Electronics [lien]

Classes

	Nom	Description
S	**Accelerometer**	Accès à l'accéléromètre MMA8453Q de la carte. Lecture de l'accélération sur les axes X, Y et Z.
S	**Button**	Accès aux 4 touches de la carte par scrutation ou par événements.
S	**Buzzer**	Accès au buzzer (déclenchement, arrêt, génération d'une notes ou d'une fréquences, réglage du volume)
S	**Color**	Représente une couleur en rouge, vert et bleu.
S	**DcMotor**	Accès à la sortie dédiée à la commande d'un moteur à courant continu 5V 200mA.
S	**Display**	Contrôle de l'afficheur couleur 1.8“ (128 px par 160 px) à ST7735R (Bus SPI) Adafruit. Texte, tracé de forme géométrique etc. **Ressource documentaire** : Les afficheurs graphiques
S	**Expansion**	Définition des entrées / sorties du connecteur d'extension.
S	**Image**	Représente une image.
S	**Legacy**	Permet d'accèder aux anciennes version de la carte.

	Nom	Description
S	**LightBulb**	Accès à la Led RVB haute luminosité.
S	**LightSensor**	Accès au capteur de luminosité (LDR).
S	**Peripherals**	Définitions des périphériques de la carte BrainPad.
S	**ServoMotor**	Accès à la sortie dédiée à la commande d'un servomoteur.
S	**TemperatureSensor**	Accès au capteur analogique de température MCP9701.
S	**TrafficLight**	Accès aux 3 Leds de la carte BrainPad.
S	**Wait**	Offre des méthodes de temporisation (en secondes ou millisecondes).

Champs

	Syntaxe	Description
	Looping	booléen toujours vrai

Exemple

exemple.cs

```
while(BrainPad.Looping){
 // Code
}
```

Méthodes

	Syntaxe	Description
S	*void* **WriteDebugMessage**(*string* message)	Affiche un message dans la fenêtre de sortie de l'éditeur Visual Studio (Community).
S	*void* **WriteDebugMessage**(*int* message)	
S	*void* **WriteDebugMessage**(*double* message)	

Exemple

exemple.cs

```
WriteDebugMessage("Température=" + temp.ToString());
```

Accelerometer

Méthodes

	Syntaxe	Description
S	*double* **ReadX**()	Lit l'accélération sur l'axe X. (-1 à 1)
S	*double* **ReadY**()	Lit l'accélération sur l'axe y. (-1 à 1)

	Syntaxe	Description
S	*double* **ReadZ**()	Lit l'accélération sur l'axe z. (-1 à 1)

Exemple

exemple.cs

```
using System.Threading;

class Program
{
    public void BrainPadSetup()
    {
    }

    public void BrainPadLoop()
    {
        BrainPad.Display.DrawText(0, 0, "X : " +
BrainPad.Accelerometer.ReadX().ToString("F3"), BrainPad.Color.White);
        BrainPad.Display.DrawText(0, 10, "Y : " +
BrainPad.Accelerometer.ReadY().ToString("F3"), BrainPad.Color.White);
        BrainPad.Display.DrawText(0, 20, "Z : " +
BrainPad.Accelerometer.ReadZ().ToString("F3"), BrainPad.Color.White);
        Thread.Sleep(100);
    }
}
}
```

Button

Méthodes

	Syntaxe	Description
S	*bool* **IsDownPressed**()	Renvoie la valeur *vrai* si le bouton **Down** a été pressé.
S	*bool* **IsLeftPressed**()	Renvoie la valeur *vrai* si le bouton **Left** a été pressé.
S	*bool* **IsPressed**()	Renvoie la valeur *vrai* si un des quatre boutons a été pressé.
S	*bool* **IsRightPressed**()	Renvoie la valeur *vrai* si le bouton **Right** a été pressé.
S	*bool* **IsUpPressed**()	Renvoie la valeur *vrai* si le bouton **Up** a été pressé.

Exemple 1 : scrutation du clavier

clavier.cs

```
class Program
{
    int increment;
```

```
    public void BrainPadSetup()
    {
        increment = 0;
    }

    public void BrainPadLoop()
    {
        if (BrainPad.Button.IsUpPressed())
        {
            increment = 100;
        }
        if (BrainPad.Button.IsDownPressed())
        {
            increment = -100;
        }
       ....
    }
}
```

Evènements

	Syntaxe	Description
	ButtonChanged	Un événement est déclenché quand l'état d'un bouton change.
	ButtonPressed	Un événement est déclenché quand l'état d'un bouton est appuyé.
	ButtonReleased	Un événement est déclenché quand l'état d'un bouton est relâché.

Enumérations

	Syntaxe	Description
	DPad	Prend les valeurs : Up, Down, Left, Right
	State	Prend les valeurs : Pressed, NotPressed

Exemple 2 : les boutons génèrent des interruptions

BPINT.cs

```
class Program
{
    public void BrainPadSetup()
    {
        BrainPad.Button.ButtonChanged += Button_ButtonChanged;
    }

    void Button_ButtonChanged(BrainPad.Button.DPad button,
BrainPad.Button.State state)
    {
        if (button == BrainPad.Button.DPad.Down)
```

```
        {
            if (state == BrainPad.Button.State.Pressed)
            {
                BrainPad.TrafficLight.TurnGreenLightOn();
            }
            else
            {
                BrainPad.TrafficLight.TurnGreenLightOff();
            }
        }
    }

    public void BrainPadLoop()
    {
        // Declared but not used
    }

}
```

Délégué

	Syntaxe	Description
	void **ButtonEventHandler**(*DPad* button, *State* state)	

Buzzer

Méthodes

	Syntaxe	Description
S	*void* **PlayFrequency**(*int* Frequency)	Joue une fréquence.
S	*void* **PlayNote**(*Note* note)	Joue une note.
S	*void* **SetVolume**(*Volume* volume)	Règle le volume.
S	*void* **Stop**()	Coupe le son.

Enumérations

	Syntaxe	Description
	Note	**A** = 880 Hz, **ASharp** = 932 Hz, **B** = 988 Hz, **C** = 1047 Hz, **CSharp** = 1109 Hz, **D** = 1175 Hz, **DSharp** = 1244 Hz, **E** = 1319 Hz, **F** = 1397 Hz, **FSharp** = 1480 Hz, **G** = 1568 Hz, **GSharp** = 1661 Hz
	Volume	**Loud** : volume par défaut, **Quiet** : plus faible

Exemple 1

frequence1.cs

```
class Program
{
```

```
    int frequency, increment;

    public void BrainPadSetup()
    {
        frequency = 0; increment = 0;
    }

    public void BrainPadLoop()
    {
        if (BrainPad.Button.IsUpPressed())
        {
            increment = 100;
        }
        if (BrainPad.Button.IsDownPressed())
        {
            increment = -100;
        }
        if (increment != 0)
        {
            frequency = frequency + increment;
            increment = 0;
            BrainPad.Buzzer.PlayFrequency(frequency);
            BrainPad.WriteDebugMessage(frequency);
            BrainPad.Wait.Seconds(0.2);
            BrainPad.Buzzer.Stop();
        }
    }
}
```

Exemple 2

frequence2.cs

```
using System.Threading;

class Program
{
    const int NoteC = 261;
    const int NoteD = 294;
    const int NoteE = 330;
    const int NoteF = 349;
    const int NoteG = 391;

    const int Whole = 1000;
    const int Half = Whole / 2;
    const int QuarterDot = Whole / 3;
    const int Quarter = Whole / 4;
    const int Eighth = Whole / 8;
```

```
    int[] note = {
                    NoteE, NoteE, NoteF, NoteG, NoteG, NoteF, NoteE,
                    NoteD, NoteC, NoteC, NoteD, NoteE, NoteE, NoteD,
                    NoteE, NoteD, NoteC, NoteC, NoteD, NoteE, NoteD,
                    NoteC, NoteC
                  };

    int[] duration = {
                    Quarter,  Quarter,  Quarter, Quarter, Quarter,
Quarter,
                    Quarter,  Quarter,  Quarter, Quarter, Quarter,
Quarter, QuarterDot,
                    Eighth, Half, Quarter,  Quarter,  Quarter, Quarter,
Quarter, Quarter,
                    Quarter,  Quarter,  Quarter, Quarter, Quarter,
Quarter, QuarterDot,
                    Eighth,Whole
                  };


    public void BrainPadSetup()
    {
    }

    public void BrainPadLoop()
    {
        for (int i = 0; i < note.Length; i++)
        {
            BrainPad.Buzzer.Stop();
            BrainPad.Buzzer.PlayFrequency(note[i]);
            Thread.Sleep(duration[i]);
        }

        Thread.Sleep(100);
    }
}
```

Color

Constructeur

	Syntaxe	Description
	Color()	Construit une nouvelle instance de la classe couleur.
	Color(*byte* red, *byte* green, *byte* blue)	Construit une nouvelle instance de la classe couleur en précisant le niveau de rouge, de vert et de bleu.

Propriétés

	Syntaxe	Description
	As565	Précise la couleur au format 565.
	B	Niveau de bleu sur un octet.
	G	Niveau de vert sur un octet.
	R	Niveau de rouge sur un octet.

Champs

	Syntaxe	Description
	Black	Couleur prédéfinie. (Color)
	Blue	Couleur prédéfinie. (Color)
	Cyan	Couleur prédéfinie. (Color)
	Green	Couleur prédéfinie. (Color)
	Magenta	Couleur prédéfinie. (Color)
	Red	Couleur prédéfinie. (Color)
	White	Couleur prédéfinie. (Color)
	Yellow	Couleur prédéfinie. (Color)

DcMotor

Méthodes

	Syntaxe	Description
S	*void* **SetSpeed**(*double* speed)	Règle la fréquence de rotation du moteur (0 à 1)
S	*void* **Stop**()	Arrête le moteur.

Display

Ressource documentaire : Les afficheurs graphiques

Méthodes

	Syntaxe	Description
S	*void* **Clear**()	Efface l'écran.
S	*void* **DrawCircle**(*int* x, *int* y, *int* r, *Color* color)	Dessine un cercle de rayon **r** à la position (**x,y**) dans la couleur **color**.
S	*void* **DrawExtraLargeLetter**(*int* x, *int* y, *char* letter, *Color* color)	Ecrit une lettre **letter** de très grande taille à la position (**x,y**) dans la couleur **color**.
S	*void* **DrawExtraLargeNumber**(*int* x, *int* y, *double* nb, *Color* color)	Ecrit un nombre **nb** de très grande taille à la position (**x,y**) dans la couleur **color**. ***Surchargée*** : *long* nb

	Syntaxe	Description
S	*void* **DrawExtraLargeText**(*int* x, *int* y, *string* text, *Color* color)	Ecrit un texte **text** de très grande taille à la position (**x,y**) dans la couleur **color**.
S	*void* **DrawFilledRectangle**(*int* x, *int* y, *int* width, *int* height, *Color* color)	Dessine un rectangle plein (**width** par **height**) à la position (**x,y**) dans la couleur **color**.
S	*void* **DrawImage**(*byte[]* data)	Dessine une image décrite par un tableau d'octets **data**. ***Surchargée*** : *int* x, *int* y, *Image* image
S	*void* **DrawLargeLetter**(*int* x, *int* y, *char* letter, *Color* color)	Ecrit une lettre de grande taille à la position (**x,y**) dans la couleur **color**.
S	*void* **DrawLargeNumber**(*int* x, *int* y, *double* nb, *Color* color)	Ecrit un nombre de grande taille à la position (**x,y**) dans la couleur **color**. ***Surchargée*** : *long* nb
S	*void* **DrawLargeText**(*int* x, *int* y, *string* text, *Color* color)	Ecrit un texte de grande taille à la position (**x,y**) dans la couleur **color**.
S	*void* **DrawLetter**(*int* x, *int* y, *char* letter, *Color* color)	Ecrit une lettre à la position (**x,y**) dans la couleur **color**.
S	*void* **DrawLine**(*int* x0, *int* y0, *int* x1, *int* y1, *Color* color)	Dessine une ligne entre les points de coordonnées (**x0,y0**) et (**x1,y1**) dans la couleur **color**.
S	*void* **DrawNumber**(*int* x, *int* y, *double* nb, *Color* color)	Ecrit un nombre à la position (**x,y**) dans la couleur **color**. ***Surchargée*** : *long* nb
S	*void* **DrawRectangle**(*int* x, *int* y, *int* width, *int* height, *Color* color)	Dessine un rectangle (**width** par **height**) à la position (**x,y**) dans la couleur **color**.
S	*void* **DrawText**(*int* x, *int* y, *string* text, *Color* color)	Ecrit un texte à la position (**x,y**) dans la couleur **color**.
S	*void* **SetPixel**(*int* x, *int* y, *Color* color)	Dessine un point à la position (**x,y**) dans la couleur **color**.
S	*void* **TurnOff**()	Eteint le rétroéclairage.
S	*void* **TurnOn**()	Active le rétroéclairage.

Expansion

Classes imbriquées

	Nom	Champs
S	AnalogInput	**PA2**, **PA3**, **PA6**, **PA7**, **PC3**
S	Gpio	**PA10**,**PA2**, **PA3**, **PA6**, **PA7**, **PA9**, **PB3**, **PB4**, **PB5**, **PC3**
S	PwmOutput	**PA10**,**PA2**, **PA3**, **PA9**

Image

Propriétés

	Syntaxe	Description
	Height	Hauteur de l'image (int)
	Pixels	Tableau de pixels (byte[])
	Width	argeur de l'image (int)

Constructeur

	Syntaxe	Description
	Image(*int* width, *int* height)	Construit une image **width** par **height**.

Méthodes

	Syntaxe	Description
	void **SetPixel**(*int* x, *int* y, *Color* color)	Fixe le pixel en (**x,y**) à la couleur **color**.

Legacy.Expansion

Classes imbriquées

	Nom	Champs
S	**AnalogInput**	**E1**, **E10**, **E**, **E2**, **E3**, **E9**
S	**Gpio**	**E1**, **E10**, **E11**, **E12**, **E2**, **E3**, **E4**, **E5**, **E6**, **E9**
S	**PwmOutput**	**E10**, **E11**, **E12**, **E9**

LightBulb

Méthodes

	Syntaxe	Description
S	*void* **SetColor**(*Color* color)	Fixe la couleur **color** de la led haute luminosité. ***Surchargée*** : *double* R, *double* G, *double* B
S	*void* **TurnOff**()	Désactive la Led.
S	*void* **TurnOn**()	Active la Led.

LightSensor

Méthodes

	Syntaxe	Description
S	*double* **ReadLightLevel**()	Lit la luminosité ambiante (0 à 1).

Exemple

LightSensor.cs

```
class Program
{
    double level;

    public void BrainPadSetup()
    {
      level = 0;
    }

    public void BrainPadLoop()
    {
        level = BrainPad.LightSensor.ReadLightLevel();
        BrainPad.WriteDebugMessage(level);
        if (level<0.5)
        {
            BrainPad.LightBulb.TurnOn();
        }
        else
        {
           BrainPad.LightBulb.TurnOff();
        }
    }
}
```

Peripherals

Champs

	Syntaxe	Description
	Buzzer	Sortie PWM pour la commande du buzzer.
	BuzzerVolumeController	Sortie PWM pour la commande du volume du buzzer.
	DcMotor	Sortie PWM pour la commande d'un moteur à CC (5V - 200mA max).
	LightSensor	Lecture du capteur de luminosité (Entrée analogique).
	ServoMotorGras	Sortie PWM pour la commande d'un servomoteur.
	TemperatureSensor	Lecture du capteur de température (entrée analogique).

Classes imbriquées

	Nom	Champs
S	**Button**	**Down**, **Left**, **Right**, **Up**
S	**DisplayGras**	**BackLight**, **ChipSelect**, **Control**, **Reset**, **SpiModule**
S	**LightBulb**	**Blue**, **Red**
S	**TouchPad**	**Left**, **Middle**, **Right**
S	**TrafficLight**	**Green**, **Red**, **Yellow**

ServoMotor

Méthodes

	Syntaxe	Description
S	*void* **SetPosition**(*int* position)	Règle la position angulaire d'un servomoteur.
S	*void* **Start**()	Active le signal de commande du servomoteur.
S	*void* **Stop**()	Désactive le signal de commande du servomoteur.

TemperatureSensor

Méthodes

	Syntaxe	Description
S	*double* **ReadTemperature**()	Lit la valeur de température.

Exemple

TemperatureSensor.cs

```
using System.Threading;

class Program
{
    public void BrainPadSetup()
    {
    }

    public void BrainPadLoop()
    {
        double tempC = BrainPad.TemperatureSensor.ReadTemperature();
        BrainPad.Display.DrawText(0, 10, "Temperature : " +
tempC.ToString("F2"), BrainPad.Color.White);
        Thread.Sleep(200);
    }
}
```

TrafficLight

Méthodes

	Syntaxe	Description
S	*void* **TurnColorOff**(*Color* color)	Désactive la led color.
S	*void* **TurnColorOn**(*Color* color)	Active la led color.
S	*void* **TurnGreenLightOff**()	Désactive la led verte.
S	*void* **TurnGreenLightOn**()	Active la led verte.
S	*void* **TurnOffAllLight**()	Désactive toutes les leds.
S	*void* **TurnRedLightOf**f()	Désactive la led rouge.
S	*void* **TurnRedLightOn**()	Active la led rouge.
S	*void* **TurnYellowLightOff**()	Désactive la led jaune.
S	*void* **TurnYellowLightOn**()	Active la led jaune.

Exemple

TrafficLight.cs

```
while (true)
        {
            BrainPad.TrafficLight.TurnGreenLightOn();
            BrainPad.Wait.Seconds(0.4);
            BrainPad.TrafficLight.TurnGreenLightOff();
            BrainPad.Wait.Seconds(0.4);
        }
```

Wait

Méthodes

	Syntaxe	Description
S	*void* **Milliseconds**(*double* milliseconds)	Déclenche une temporisation exprimée en millisecondes.
S	*void* **Seconds**(*double* seconds)	Déclenche une temporisation exprimée en secondes.

Exemple

TrafficLight.cs

```
while (true)
        {
            BrainPad.TrafficLight.TurnGreenLightOn();
            BrainPad.Wait.Seconds(0.4);
            BrainPad.TrafficLight.TurnGreenLightOff();
            BrainPad.Wait.Seconds(0.4);
        }
```

Pour aller plus loin

Le connecteur mikroBUS permet d'étendre les fonctionnalités de la carte BrainPad (GPIO, I2C, SPI, UART).
Voir la page CH5c. Exemples codés en C# pour la carte BrainPad v1.