Arduino et sheild EasyVR

Doc en anglais

Ci-dessous une traduction“Google” avec ses défauts

EasyVR est un Module de reconnaissance vocale conçu pour ajouter facilement des fonctions polyvalentes,des capacités de reconnaissance vocale efficaces à pratiquement n'importe quelle application. Le module EasyVR peut être utilisé avec n'importe quel hôte avec une interface UART (Serie RS232) alimentée en 3.3V-5V, comme PIC, Arduino, Raspberry. Certains exemples d'application incluent l'automatisation domotique, telle que la lumière commandée par la voix, Interrupteurs, serrures ou lits, ou ajouter de la reconnaissance vocale aux robots.

EasyVR Caractéristiques :

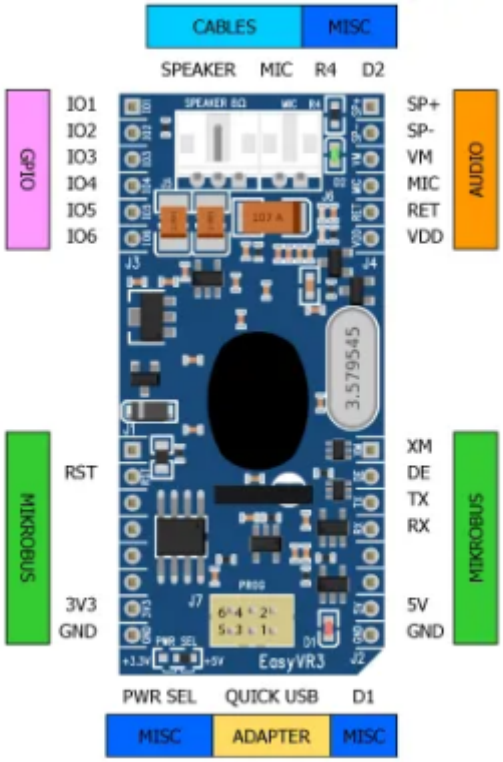
Dans les commandes Speaker Independent (SI) pour les commandes de base prêtes pour les Langues suivantes:

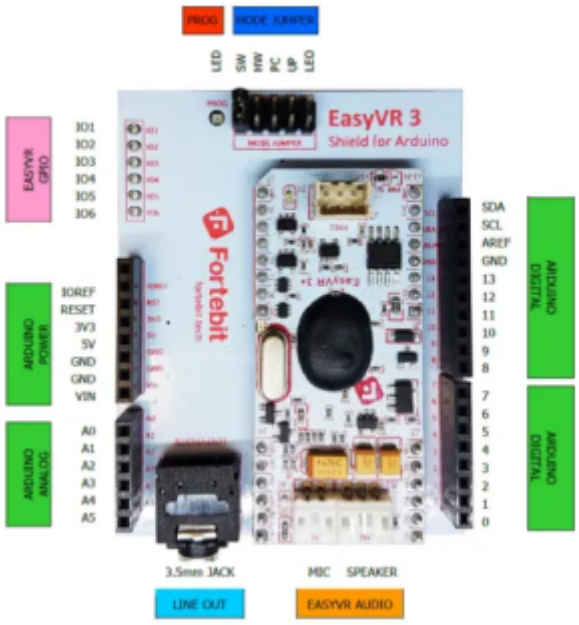
- Anglais (États-Unis)
- Italien
- Allemand
- **Français**
- Espagnol
- Japonais

Prise en charge jusqu'à 32 utilisateurs

- - Commandes déclenchés à la voix ou par evenements.
- - Les mots de passe. Les commandes personnalisées SD peuvent être prononcées dans n'importe quelle langue.
- - Utilisation et une interface utilisateur graphique simple pour programmer des commandes vocales et audio.
- - Module peut être utilisé avec n'importe quel hôte avec une interface UART (alimenté à 3.3V - 5V)
- - Simple et robuste protocole sériel documenté pour accéder et programmer via la carte hôte 6 lignes GPIO (IO1, IO2, IO3, IO4, IO5, IO6) pouvant être commandées par de nouvelles commandes de protocole.
- - PWM qui prend en charge les haut-parleurs 8Ω.
- - Lecture sonore d'un maximum de 9 minutes de sons

Schémas technique





Interface série

L'EasyVR est un module “esclave” communiquant via une interface série asynchrone (communément appelée Interface UART), avec Les caractéristiques suivantes:

- Taux de transmission: 9600 (Par défaut), 19200, 38700, 57600, 115200
- Cadre: 8 Les bits de données, N O parité, 1 Bit d'arrêt
- La ligne de données d'entrée du récepteur est ERX, alors que la ligne de données de sortie de l'émetteur est ETX. Aucune ligne de contrôle n'est utilisée.

Exemple d'une série Cadre de données représentant le caractère “A” (décimal 65 ou hexadécimal 41)

Microphone

Le microphone fourni avec le module EasyVR est un microphone à condensateur électreté omnidirectionnel (Horn EM9745P-382):

- Sensibilité- 38dB (0dB = 1V / Pa à 1KHz)
- Impédance de charge 2.2K
- Tension de fonctionnement 3V
- Réponse en fréquence presque plate dans la plage 100Hz-20 kHz
- Si vous utilisez un microphone avec des spécifications différentes, la précision de la reconnaissance peut être affectée négativement

Une bonne fixation du micro peut nettement améliorer la qualité de la reconnaissance vocale

1. Montage encastré -L'élément de microphone doit être positionné aussi près du support surface et doit être bien maintenu dans un boîtier en plastique. Il ne doit pas y avoir d'espace entre le microphone et le boîtier. Un tel espace peut conduire à une résonance acoustique, Ce qui peut réduire la précision de la reconnaissance.

2. Aucun obstacle, avec un grand trou - La zone devant l'élément de microphone doit être dégagée sans obstruction, afin d'éviter toute interférence avec la reconnaissance. Le diamètre du trou dans le boîtier pour l'entrée du micro, doit être d'au moins 5 mm. Toute la surface en plastique devant le micro doit être aussi mince que possible, ne dépassant pas 0,7 mm.

3.Isolation - Le micro doit être isolé acoustiquement du boîtier. Ceci peut être réalisé en entourant l'élément de micro avec un matériau spongieux tel que du caoutchouc ou de la mousse. Le micro fourni possède ce type de mousse isolante. Le but est de prévenir les bruits parasites produits par la manipulation,de tels bruits parasites peuvent réduire la précision de la reconnaissance.

4.Distance -Si le micro est placé entre 15 cm à 30 cm de la bouche du locuteur, La puissance du signal diminue d'un facteur quatre en fonction de la distance. Les différences entre une voix forte et une voix douce peuvent également faire varier d'un facteur quatre la puissance du signal. Bien que le préamplificateur interne EasyVR compense une large plage dynamique de la puissance du signal d'entrée, si sa portée est dépassée, l'application utilisateur peut fournir une réponse au locuteur au sujet du volume de la voix. voir annexe “Erreur R codes”

-1--2--3-

EasyVR Shield 3 pour Arduino

Parametres

Sur le côté inférieur de la carte il y a deux résistances SMD que vous pouvez déplacer pour sélectionner les deux broches de Arduino sur lequel l'EasyVR sera connecté en mode Serial Logiciel (Mode Jumper sur SW).

RX-Broche du récepteur série Serial

- D12-Utiliser la broche numérique 12 comme récepteur série (par défaut)
- D8-Utiliser la broche numérique 8 comme récepteur série

TX-Broche du transmetteur série logiciel

- D13-Utiliser la broche numérique 13 comme émetteur série (par défaut)
- D9-Utiliser la broche numérique 9 comme émetteur série

Le choix des broches 12 - 13 pour une compatibilité ascendante avec les révisions matérielles précédentes du bouclier EasyVR. Cependant, ces broches peuvent également être utilisées pour l'interface SPI, donc un autre choix de broches 8 - 9 est prévu. Si vous voulez utiliser des broches différentes assurez-vous La broche du récepteur supporte les interruptions de changement.

Réglages

On peut faire des réglages pour améliorer la reconnaissance vocale dans le menu “Tools” et “Recognition Settings”

- “Stricness Control” ou Niveau de Contrôle Rigoureux , après essai le placer entre “Easy” et “Normal” (2)
- “Confidence Threshold” ou Seuil de confiance Réglage à 1 entre “Looser” et “Typical”
- “Microphone Distance” ou Distance du Micro , nous l'avons reglé à 30 cm , cela permet d'enlever tout les sons qui sont loin , mais il faut s'approcher pour parler , Les mots sont mieux reconnus

Mais cela reste quand même assez pointu pour faire un bon réglage , je vais me contenter dans un premier temps des sons enregistrés en implicite (voir WordSet 1,2,et 3). Ils sont mieux reconnu même si l'on change de voix.

Démarrage rapide avec un Arduino Uno

- 1.Insérez le Shield EasyVR sur la carte Arduino
- 2.Si vous voulez une sortie audio, Connecté un Haut-parleur 8Ω sur le Connecteur J5 sur le Module EasyVR ou connecter un casque ou des haut-parleurs amplifiés au LINE OUT 3.5mm Prise audio sur le sheild
- 3.Connectez le microphone fourni à MIC Connecteur J6) Sur le module EasyVR
- 4.Installer l'EasyVR Bibliothèques Arduino Sur votre PC (Détails à Http://arduino.cc/fr/Guide/Libraries)

- 5.Connectez votre carte Arduino à votre PC via USB

Avec Arduino 2009 Uno - Mega

Testez le Shield avec Arduino

- 1. Réglez le cavalier Mode (J7) en position SW
- 2. Ouvrez l'exemple d'esquisse TestEasyVR à partir de votre menu IDE “Fichier ”>“ Exemples ”>“ EasyVR ”
- 3. Téléchargez l'esquisse et ouvrez la fenêtre “Serial Monitor”
- 4. Voir les commentaires en haut de l'esquisse pour les détails d'utilisation

Testez le bouclier avec le EasyVR Commander

- 1.Assurez-vous que le cavalier Mode (J7) est en position PC
- 2.Ouvrez le EasyVR Commander et connecté au même port série utilisé par Arduino

Télécharger un nouveau son dans le tableau ou mise à jour du firmware

- 1. Assurez-vous que le cavalier Mode (J7) est en position UP
- 2. Ouvrez EasyVR Commander et sélectionnez le port série Arduino
- 3. Lorsque vous êtes déconnecté, Mettre à jour les données personnalisées “ du ” Fichier “(Ou” Mise à jour du firmware “ du ” Aidez-moi “Menu)

Lorsque le EasyVR Commander est connecté, vous pouvez générer un modèle de code pour Arduino Cela va Utiliser les bibliothèques fournies (voir Librairie EasyVR pour Arduino). Il vous suffit d'écrire des commandes reconnues.

les programmes

EasyVR-Commander-3.14.0.232-All

Captures écrans

Capture écrans EasyVR3

EasyVR-Arduino-library-1.9.1.zip

Le programme EasyVR V3.11.0 est utilisé pour entrer les commandes vocales dans le module. Ce programme est un programme Windows, on peut l'installer sous Linux via wine , mais il faut paramétrer les ports com 1, com2 ou autres.

Une méthode, dans le répertoire **.wine/dosdevices** taper la commande **ln -s /dev/ttyUSB0 com1** , cela ne fonctionne pas à tout les coups…

Lancer le logiciel EasyVR3 et connecté votre Arduino avec le shield inséré et vérifié le cavalier sur “PC” , connecté par “File” et “Connect” . Vous avez toutes les commandes vocales qui s'affichent dans les différents groupes. Au départ il n'y a que dans le groupe “Trigger” avec “Robot” et dans les groupes “Wordset”.

Il est proposé 15 groupes pour la programmation des commandes. Avec 32 commandes Maxi à configurer en tout. La premier chose à faire c'est de configurer le Français, dans “Tools” → “Set Language” → “French”. Ceci pour les commandes implicites en Francais , mais pas pour les menus...

On va faire un exemple simple pour commencer , allumer une LED sur la broche 9 de l'Arduino avec 2 commandes vocales : ALLUME et ETEINT

On se positionne sur le Groupe 1 (Index 1) , on va dans “Edit” → “Add Command” ou ” Shift+INS“. Une nouvelle ligne s'affiche dans la fenêtre de droite, taper dans le champ colonne “Label” le mot “ALLUME”. En fait on peut taper ce que l'on veut, mais si l'on tape “ETEINT” et que l'on veut allumer cela être très compliqué à gérer. Le logiciel ne reconnais pas le mot mais le son que l'on va enregistrer pour ce mot.

Maintenant on va enregistrer le son lié au mot, Sélectionner la ligne “ALLUME” dans “Edit” → “Train Command” ou “ENTER” Un pop up s'affiche nous demandant de cliquer sur le bouton “Phase1” . A partir de ce clic, l'affichage “Speak now...” dure 5 secondes pour enregistre le son. S'il n'y pas pas reconnaissance d'un son une fenêtre “Timeout expired” s'affiche . Cliquer sur “OK” et recommencer “Phase 1”. Quand la phase 1 est terminée , une phase 2 et proposée pour vérifier à nouveau le son , on clique sur “Phase 2” et on réenregistre le même son, si les 2 sons ne sont pas identique il faut recommencer la phase 2 et si le délai est trop long ou le son trop long , il faut tout recommencer au début. Quand le son est bon , on a dans le champ “Trained” la valeur 2 , qui indique que les 2 enregistrements sont bon et prêt à être utilisés . Si dans le champ “Conflict”, OK n'est pas inscrit, il faut recommencer tout l'enregistrement du mot.

On procède de même pour le deuxième mot et tous les mots que l'on veut enregistrer (MAX 32)

On insère également dans le groupe “Trigger” le mot “Arduino” que l'on enregistre de la même manière. Ceci pour avoir une sorte de mot de passe pour démarrer la reconnaissance vocale. On pourrait mettre n'importe quel mot , il faut juste s”en souvenir...

Quand les mots/sons sont enregistrés, on va générer le code pour l'Arduino. Dans “File” → “Generate Code” , on choisi le répertoire et le nom du fichier à généré et “Enregistrer”

On se déconnecte : “File” → “Disconnect”, On remet le cavalier “J7” sur la position “SW”

On ouvre l'IDE Arduino, on charge le programme que l'on vient de générer, on vérifie le type de carte et le port Com utilisé.

dans “void setup()” du programme on insère les lignes :

setuppin9.c

```
pinMode(9,OUTPUT);
digitalWrite(9,LOW);
```

Dans le même programme , on va dans le sous programme “void action()”

dans la partie “Case GO_ARDUINO” on insère la ligne

allergroup1.c

```
group = GROUP_1;
```

dans la partie “Case GROUP_1” on insere :

GROUP_1.c

```
switch (idx)
{
case G1_ALLUME:
  digitalWrite(9, HIGHT);
  group = GROUP_1; // on peut mettre group = GROUP_0 si l'on veut à
chaque fois redire le mot de passe "Arduino"
  break;
case G1_ETEINT:
  digtalWrite(9, LOW);
  group = GROUP_1;
  break;
}
```

Le code complet pour allumer la LED 9 et l’éteindre par reconnaissance vocale

LED9.c

```
#include "Arduino.h"
#if !defined(SERIAL_PORT_MONITOR)
  #error "Arduino version not supported. Please update your IDE to the
latest version."
#endif

#if defined(SERIAL_PORT_USBVIRTUAL)
  // Shield Jumper on HW (for Leonardo and Due)
  #define port SERIAL_PORT_HARDWARE
  #define pcSerial SERIAL_PORT_USBVIRTUAL
#else
  // Shield Jumper on SW (using pins 12/13 or 8/9 as RX/TX)
  #include "SoftwareSerial.h"
  SoftwareSerial port(12, 13);
  #define pcSerial SERIAL_PORT_MONITOR
#endif

#include "EasyVR.h"

EasyVR easyvr(port);
```

```
//Groups and Commands
enum Groups
{
  GROUP_0  = 0,
  GROUP_1  = 1,
 };

enum Group0
{
  G0_ARDUINO = 0,
};

enum Group1
{
  G1_ALLUME = 0,
  G1_ETEINT = 1,
};

int8_t group, idx;

void setup()
{
  pinMode(9, OUTPUT);
  digitalWrite(9,LOW);

  // setup PC serial port
  pcSerial.begin(9600);

  // bridge mode?
  int mode = easyvr.bridgeRequested(pcSerial);
  switch (mode)
  {
  case EasyVR::BRIDGE_NONE:
    // setup EasyVR serial port
    port.begin(9600);
    // run normally
    pcSerial.println(F("---"));
    pcSerial.println(F("Bridge not started!"));
    break;

  case EasyVR::BRIDGE_NORMAL:
    // setup EasyVR serial port (low speed)
    port.begin(9600);
    // soft-connect the two serial ports (PC and EasyVR)
    easyvr.bridgeLoop(pcSerial);
    // resume normally if aborted
    pcSerial.println(F("---"));
    pcSerial.println(F("Bridge connection aborted!"));
```

```
    break;

  case EasyVR::BRIDGE_BOOT:
    // setup EasyVR serial port (high speed)
    port.begin(115200);
    // soft-connect the two serial ports (PC and EasyVR)
    easyvr.bridgeLoop(pcSerial);
    // resume normally if aborted
    pcSerial.println(F("---"));
    pcSerial.println(F("Bridge connection aborted!"));
    break;
  }

  while (!easyvr.detect())
  {
    Serial.println("EasyVR not detected!");
    delay(1000);
  }

  easyvr.setPinOutput(EasyVR::IO1, LOW);
  Serial.println("EasyVR detected!");
  easyvr.setTimeout(10);
  easyvr.setLanguage(5);

  group = EasyVR::TRIGGER; //<-- start group (customize)
}

void action();

void loop()
{
  if (easyvr.getID() < EasyVR::EASYVR3)
    easyvr.setPinOutput(EasyVR::IO1, HIGH); // LED on (listening)

  Serial.print("Say a command in Group ");
  Serial.println(group);
  easyvr.recognizeCommand(group);

  do
  {
    // can do some processing while waiting for a spoken command
  }
  while (!easyvr.hasFinished());

  if (easyvr.getID() < EasyVR::EASYVR3)
    easyvr.setPinOutput(EasyVR::IO1, LOW); // LED off

  idx = easyvr.getWord();
  if (idx >= 0)
  {
    // built-in trigger (ROBOT)
```

```
    // group = GROUP_X; <-- jump to another group X
    return;
  }
  idx = easyvr.getCommand();
  if (idx >= 0)
  {
    // print debug message
    uint8_t train = 0;
    char name[32];
    Serial.print("Command: ");
    Serial.print(idx);
    if (easyvr.dumpCommand(group, idx, name, train))
    {
      Serial.print(" = ");
      Serial.println(name);
    }
    else
      Serial.println();
    // beep
    easyvr.playSound(0, EasyVR::VOL_FULL);
    // perform some action
    action();
  }
  else // errors or timeout
  {
    if (easyvr.isTimeout())
      Serial.println("Timed out, try again...");
    int16_t err = easyvr.getError();
    if (err >= 0)
    {
      Serial.print("Error ");
      Serial.println(err, HEX);
    }
  }
}

void action()
{
    switch (group)
    {
    case GROUP_0:
      switch (idx)
      {
      case G0_ARDUINO:
       group = GROUP_1;// apres avoir dit le mot "Arduino" on va au
groupe 1
        break;
      }
      break;
```

```
    case GROUP_1:
      switch (idx)
      {
      case G1_ALLUME:
        digitalWrite(9, HIGH);
        group = GROUP_1;// on reste dans le groupe 1
        break;
      case G1_ETEINT:
        digitalWrite(9, LOW);
        group = GROUP_1;// on reste dans le groupe 1
        break;
      }
      break;
    }
}
```

le code qui commande l'allumage de la LED 9 par le miaulement d'un chat ou par le meuglement d'une vache et qui l'éteint par le barrissement d'un éléphant. Le code de démarrage est toujours "Arduino"

les sons

Les sons Miaou Meuh et Elephant

meuhhh.c

```
#include "Arduino.h"
#if !defined(SERIAL_PORT_MONITOR)
  #error "Arduino version not supported. Please update your IDE to the
latest version."
#endif

#if defined(SERIAL_PORT_USBVIRTUAL)
  // Shield Jumper on HW (for Leonardo and Due)
  #define port SERIAL_PORT_HARDWARE
  #define pcSerial SERIAL_PORT_USBVIRTUAL
#else
  // Shield Jumper on SW (using pins 12/13 or 8/9 as RX/TX)
  #include "SoftwareSerial.h"
  SoftwareSerial port(12, 13); // on déclare le port série virtuel sur
les broches 12 et 13 pour le dialogue entre le Shield et l'Arduino
  #define pcSerial SERIAL_PORT_MONITOR
#endif

#include "EasyVR.h"  // on inclue la bibliothèque EasyVR

EasyVR easyvr(port);
```

```
//Groups and Commands
enum Groups  // les sons sont programmés avec le logiciel EasyVR v3.11.0
{
  GROUP_0  = 0,// le groupe 0 contient le mot de passe de démarrage "Arduino" sans ce mot de passe le shield ne démarre pas la reconnaissance vocale
  GROUP_1  = 1, // les 2 sons pour "ALLUME" et "ÉTEINT"
  GROUP_2  = 2, // Les 2 sons pour allumer la LED avec une tape sur le bureau et 2 Tapes pour éteindre la LED
  GROUP_3  = 3, // les 3 sons : Miau et Meuhh pour allumer , le barrissement pour éteindre
  // GROUP_0 correspond à l'index du groupe  0 dans le logiciel EasyVR etc ...
};

enum Group0
{
  G0_ARDUINO = 0,// le son "ARDUINO" correspond à l'index du son  0 dans le groupe 0
};

enum Group1
{
  G1_ALLUME = 0,// le son "ALLUME" correspond à l'index du son  0 dans le groupe 1
  G1_ETEINT = 1,// le son "ETEINT" correspond à l'index du son  1 dans le groupe 1
};

enum Group2
{
  G2_TAP = 0,  // le son "TAP" correspond à l'index du sons  0 dans le groupe 2
  G2_TAPTAP = 1,// le son "TAPTAP" correspond à l'index du son  1 dans le groupe 2
};

enum Group3
{
  G3_MIAOU = 0,  // le son "MIAOU" correspond à l'index du son  0 dans le groupe 3
  G3_MEUHHH = 1,  // le son "MEUHH" correspond à l'index du son  1 dans le groupe 3
  G3_ELEPHANT = 2, // le son "BARRISSEMENT" correspond à l'index du son 2 dans le groupe 3
};
```

```
int8_t group, idx;// Déclare les variables "group" et "idx" en entier
signé sur 8 bits

void setup()
{
  pinMode(9, OUTPUT); // on declare la broche 9 de l'Arduino en sortie
  digitalWrite(9,LOW); // on eteint la LED 9

  // setup PC serial port
  pcSerial.begin(9600);

  // bridge mode?
  int mode = easyvr.bridgeRequested(pcSerial);
  switch (mode)
  {
  case EasyVR::BRIDGE_NONE:
    // setup EasyVR serial port
    port.begin(9600);
    // run normally
    pcSerial.println(F("---"));
    pcSerial.println(F("Bridge not started!"));
    break;

  case EasyVR::BRIDGE_NORMAL:
    // setup EasyVR serial port (low speed)
    port.begin(9600);
    // soft-connect the two serial ports (PC and EasyVR)
    easyvr.bridgeLoop(pcSerial);
    // resume normally if aborted
    pcSerial.println(F("---"));
    pcSerial.println(F("Bridge connection aborted!"));
    break;

  case EasyVR::BRIDGE_BOOT:
    // setup EasyVR serial port (high speed)
    port.begin(115200);
    // soft-connect the two serial ports (PC and EasyVR)
    easyvr.bridgeLoop(pcSerial);
    // resume normally if aborted
    pcSerial.println(F("---"));
    pcSerial.println(F("Bridge connection aborted!"));
    break;
  }

  while (!easyvr.detect())
  {
    Serial.println("EasyVR not detected!");
    delay(1000);
  }

  easyvr.setPinOutput(EasyVR::IO1, LOW); // on éteint la led IO1 sur le
```

```
module reconnaissance vocale
  Serial.println("EasyVR detected!");
  easyvr.setTimeout(10);
  easyvr.setLanguage(5);

  group = EasyVR::TRIGGER; //<-- start group (customize) le groupe du
début ici on demande le mot de passe "ARDUINO"
}

void action(); // on test si on reconnais un son

void loop()
{
  if (easyvr.getID() < EasyVR::EASYVR3)
    easyvr.setPinOutput(EasyVR::IO1, HIGH); // LED on (listening)

  Serial.print("Say a command in Group ");
  Serial.println(group);
  easyvr.recognizeCommand(group);

  do
  {
    // can do some processing while waiting for a spoken command
  }
  while (!easyvr.hasFinished());

  if (easyvr.getID() < EasyVR::EASYVR3)
    easyvr.setPinOutput(EasyVR::IO1, LOW); // LED off

  idx = easyvr.getWord();
  if (idx >= 0)
  {
    // built-in trigger (ROBOT)
    // group = GROUP_X; <-- jump to another group X
    return;
  }
  idx = easyvr.getCommand();
  if (idx >= 0)
  {
    // print debug message
    uint8_t train = 0;
    char name[32];
    Serial.print("Command: ");
    Serial.print(idx);
    if (easyvr.dumpCommand(group, idx, name, train))
    {
      Serial.print(" = ");
      Serial.println(name);
    }
```

```
    else
      Serial.println();
    // beep
    easyvr.playSound(0, EasyVR::VOL_FULL);
    // perform some action
    action();
  }
  else // errors or timeout
  {
    if (easyvr.isTimeout())
      Serial.println("Timed out, try again...");
    int16_t err = easyvr.getError();
    if (err >= 0)
    {
      Serial.print("Error ");
      Serial.println(err, HEX);
    }
  }
}

void action()
{
    switch (group)
    {
    case GROUP_0:
      switch (idx)
      {
      case G0_ARDUINO:
        // write your action code here
        // group = GROUP_X; <-- or jump to another group X for
composite commands
       group = GROUP_3;
        break;
      }
      break;
    case GROUP_1:
      switch (idx)
      {
      case G1_ALLUME:
        digitalWrite(9, HIGH);
        group = GROUP_1;
        break;
      case G1_ETEINT:
        digitalWrite(9, LOW);
        group = GROUP_2;
        break;
      }
      break;
    case GROUP_2:
      switch (idx)
      {
```

```
      case G2_TAP:
         digitalWrite(9, HIGH);
        group = GROUP_2;
        break;
      case G2_TAPTAP:
        digitalWrite(9, LOW);
        group = GROUP_3;
        break;
      }
      break;
    case GROUP_3:
      switch (idx)
      {
      case G3_MIAOU:
        digitalWrite(9, HIGH);
        group = GROUP_3;
        break;
      case G3_MEUHHH:
        digitalWrite(9, HIGH);
        group = GROUP_3;
        break;
      case G3_ELEPHANT:
        digitalWrite(9, LOW);
        group = GROUP_3;
        break;
      }
      break;
    }
}
```

les commandes en mode implicite

Le programme pour allumer/éteindre la LED9 , en parlant Français , d'abord dire le mot “ROBOT” , un bip retenti. Ensuite le mot “ACTION” allume le LED9 et le mot “ARRETE” éteint la LED9.

ROBOTLED9.cpp

```
#include "Arduino.h"
#if !defined(SERIAL_PORT_MONITOR)
  #error "Arduino version not supported. Please update your IDE to the
latest version."
#endif

#if defined(SERIAL_PORT_USBVIRTUAL)
  // Shield Jumper on HW (for Leonardo and Due)
```

```
  #define port SERIAL_PORT_HARDWARE
  #define pcSerial SERIAL_PORT_USBVIRTUAL
#else
  // Shield Jumper on SW (using pins 12/13 or 8/9 as RX/TX)
  #include "SoftwareSerial.h"
  SoftwareSerial port(12, 13);
  #define pcSerial SERIAL_PORT_MONITOR
#endif

#include "EasyVR.h"

EasyVR easyvr(port);


int8_t group, idx;

void setup()
{

  pinMode(9, OUTPUT);
  digitalWrite(9,LOW);

  // setup PC serial port
  pcSerial.begin(9600);

  // bridge mode?
  int mode = easyvr.bridgeRequested(pcSerial);
  switch (mode)
  {
  case EasyVR::BRIDGE_NONE:
    // setup EasyVR serial port
    port.begin(9600);
    // run normally
    pcSerial.println(F("---"));
    pcSerial.println(F("Bridge not started!"));
    break;

  case EasyVR::BRIDGE_NORMAL:
    // setup EasyVR serial port (low speed)
    port.begin(9600);
    // soft-connect the two serial ports (PC and EasyVR)
    easyvr.bridgeLoop(pcSerial);
    // resume normally if aborted
    pcSerial.println(F("---"));
    pcSerial.println(F("Bridge connection aborted!"));
    break;

  case EasyVR::BRIDGE_BOOT:
    // setup EasyVR serial port (high speed)
    port.begin(115200);
    // soft-connect the two serial ports (PC and EasyVR)
```

```
    easyvr.bridgeLoop(pcSerial);
    // resume normally if aborted
    pcSerial.println(F("---"));
    pcSerial.println(F("Bridge connection aborted!"));
    break;
  }

  while (!easyvr.detect())
  {
    Serial.println("EasyVR not detected!");
    delay(1000);
  }

  easyvr.setPinOutput(EasyVR::IO1, LOW);
  Serial.println("EasyVR detected!");
  easyvr.setTimeout(5);
  easyvr.setLanguage(5); // 0 = Anglais; 5 = Français, en fait le son
est Français mais le texte reste Anglais

  group = 0; //<-- start group (considering to start from custom
grammar 0)
}

void action();

void loop()
{
  if (easyvr.getID() < EasyVR::EASYVR3)
    easyvr.setPinOutput(EasyVR::IO1, HIGH); // LED on (listening)

  Serial.print(" Dire la commande WS ");
  Serial.println(group);
  easyvr.recognizeWord(group); // pou les commandes SI ( Multi
Locuteur)
  //easyvr.recognizeCommand(group); // used for SD commands/trigger or
for built-in trigger word

  do
  {
    // can do some processing while waiting for a spoken command
  }
  while (!easyvr.hasFinished());

  if (easyvr.getID() < EasyVR::EASYVR3)
    easyvr.setPinOutput(EasyVR::IO1, LOW); // LED off

  /* This part of the code is not needed if using SI commands only
(excluding the built-in trigger ROBOT)
  if (idx >= 0)
```

```
  {
    // built-in trigger (ROBOT)
    // group = GROUP_X; <-- jump to another group X
    return;
  }
  idx = easyvr.getCommand();
  */

  idx = easyvr.getWord();
  if (idx >= 0)
  {
    // print debug message for SI commands
    uint8_t flags, num;
    char name[32];
    Serial.print("Command: ");
    Serial.print(idx);
    if (easyvr.dumpGrammar(group, flags, num))
    for (int8_t i = 0; i <= idx; ++i) easyvr.getNextWordLabel(name);
    pcSerial.print(F(" = "));
    pcSerial.println(name);

    // beep
    easyvr.playSound(0, EasyVR::VOL_FULL);
    // perform some action
    action();
  }
  else // errors or timeout
  {
    if (easyvr.isTimeout())
      Serial.println("Timed out, try again...");
    int16_t err = easyvr.getError();
    if (err >= 0)
    {
      Serial.print("Error ");
      Serial.println(err, HEX);
    }
  }
}

void action()
{
    switch (group)
    {
    case 0: // groupe TRIGGER
      switch (idx)
      {
      case 0:

        group = 1; // saute au Groupe WORDSET N°1
        break;
      }
```

```
      break;
    case 1: // Groupe WORDSET N°1
      switch (idx)
      {
      case 0:
        digitalWrite(9, HIGH);
        group = 1;
        break;

      case 6:
        digitalWrite(9, LOW);
        group = 1;
        break;
      }
      break;
       }
}
```

Programme de commande d'une LED et d'un servo-moteur par reconnaisance vocale

lumiere.ino

```
#if defined(ARDUINO) && ARDUINO >= 100
  #include "Arduino.h"
  #include "SoftwareSerial.h"
  SoftwareSerial port(12,13);
#else // Arduino 0022 - use modified NewSoftSerial
  #include "WProgram.h"
  #include "NewSoftSerial.h"
  NewSoftSerial port(12,13);
#endif

#include "EasyVR.h"
EasyVR easyvr(port);

//Groups and Commands
enum Groups
{
  GROUP_0  = 0,
  GROUP_1  = 1,
  GROUP_2  = 2,
  GROUP_3  = 3,
};

enum Group0
{
```

```
  G0_ADAM = 0,
};

enum Group1
{
  G1_VEILLE = 0,
  G1_LUMIERE = 1,
};

enum Group2
{
  G2_ACTIVATION = 0,
};

enum Group3
{
  G3_OUI = 0,
  G3_NON = 1,
};


EasyVRBridge bridge;

int8_t group, idx, LED;


void setup()
{
  // bridge mode?
  if (bridge.check())
  {
    cli();
    bridge.loop(0, 1, 12, 13);
  }
  // run normally
  Serial.begin(9600);
  port.begin(9600);

  if (!easyvr.detect())
  {
    Serial.println("EasyVR not detected!");
    for (;;);
  }
  enum Level {NORMAL};
  easyvr.setPinOutput(EasyVR::IO1, LOW);
  Serial.println("EasyVR detected!");
  easyvr.setTimeout(10);
  easyvr.setLanguage(5);
  easyvr.setKnob(0);
  easyvr.setLevel(2);
```

```
  group = EasyVR::TRIGGER; //<-- start group (customize)
}

void action();

void loop()
{
  easyvr.setPinOutput(EasyVR::IO1, HIGH); // LED on (listening)

  Serial.print("Say a command in Group ");
  Serial.println(group);
  easyvr.recognizeCommand(group);

  do
  {
    // can do some processing while waiting for a spoken command
  }
  while (!easyvr.hasFinished());

  easyvr.setPinOutput(EasyVR::IO1, LOW); // LED off

  idx = easyvr.getWord();
  if (idx >= 0)
  {
    // built-in trigger (ROBOT)
    // group = GROUP_X; <-- jump to another group X
    return;
  }
  idx = easyvr.getCommand();
  if (idx >= 0)
  {
    // print debug message
    uint8_t train = 0;
    char name[32];
    Serial.print("Command: ");
    Serial.print(idx);
    if (easyvr.dumpCommand(group, idx, name, train))
    {
      Serial.print(" = ");
      Serial.println(name);
    }
    else
      Serial.println();

    // perform some action
    action();
  }
  else // errors or timeout
```

```
  {
    if (easyvr.isTimeout())
      Serial.println("Timed out, try again...");
    int16_t err = easyvr.getError();
    if (err >= 0)
    {
      Serial.print("Error ");
      Serial.println(err, HEX);
    }
  }
}

void action()
{
    switch (group)
    {
    case GROUP_0:
      switch (idx)
      {
      case G0_ADAM:
        easyvr.playSound(7, EasyVR::VOL_FULL);
        group = GROUP_1;
        break;
      }
      break;
    case GROUP_1:
      switch (idx)
      {
      case G1_VEILLE:
        if (LED == 1)
        {
         easyvr.playSound(3, EasyVR::VOL_FULL);
         group = GROUP_3;
        }
        else
        {
         easyvr.playSound(1, EasyVR::VOL_FULL);
         group = GROUP_2;
        }
        break;
      case G1_LUMIERE:
      if (LED == 1)
      {
        LED = 0;
        easyvr.setPinOutput(EasyVR::IO2, LOW);
        easyvr.playSound(9, EasyVR::VOL_FULL);
        group = GROUP_1;
      }
      else
      {
        LED = 1;
```

```
        easyvr.setPinOutput(EasyVR::IO2, HIGH);
        easyvr.playSound(8, EasyVR::VOL_FULL);
        group = GROUP_1;
      }
        break;
      }
      break;
    case GROUP_2:
      switch (idx)
      {
      case G2_ACTIVATION:
        easyvr.playSound(2, EasyVR::VOL_FULL);
        group = GROUP_0;
        break;
      }
      break;
    case GROUP_3:
      switch (idx)
      {
      case G3_OUI:
        LED = 0;
        easyvr.playSound(1, EasyVR::VOL_FULL);
        easyvr.setPinOutput(EasyVR::IO2, LOW);
        group = GROUP_2;
        break;
      case G3_NON:
        easyvr.playSound(1, EasyVR::VOL_FULL);
        group = GROUP_2;
        break;
      }
      break;
    }
}
```

Explications

Voici le code, explication plus en détails :

```
easyvr.playSound(1, EasyVR::VOL_FULL);
```

Cette ligne a pour fonction de sélectionner un enregistrement préalablement fait sur Audacity (ou autre) lorsqu'une instruction aura été reconnue.

Cet enregistrement est à envoyer sur la carte à l'aide de QuickSynthesis, un logiciel complémentaire du logiciel EasyVRCommander (téléchargeable sur ce site avec le manuel utilisateur et la datasheet pour de plus amples informations : http://www.veear.eu/products/easyvr-arduino-shield/)

qui permet d'enregistrer les commandes vocales à reconnaître à l'aide du micro fourni avec le shield.

Si on décortique ce qu'il y a entre parenthèses, le 1 signifie que c'est le premier enregistrement qui est sélectionné (Dans mon cas, il dira “Au revoir Monsieur) dans la liste de tout ceux que vous aurez enregistré sur la carte. VOL_FULL sert évidemment à contrôler le volume sortant des haut-parleurs.

```
case G1_LUMIERE:
      if (LED == 1)
```

Plus de détails sur ces deux lignes : La première ligne correspond à l'instruction reconnue par le micro (petite info, une fois les commandes enregistrées sur EasyVRCommander, il vous est possible de générer automatiquement le code, il ne vous suffira alors plus qu'à le modifier à votre guise pour répondre à vos besoins), celle du dessous correspond à une condition qui détecte si la LED est déjà allumée ou pas, l'utilité étant de n'utiliser que le mot LUMIERE pour allumer et éteindre la LED.

Les Liens

Le site VeeaR

Github, EasyVR et Arduino