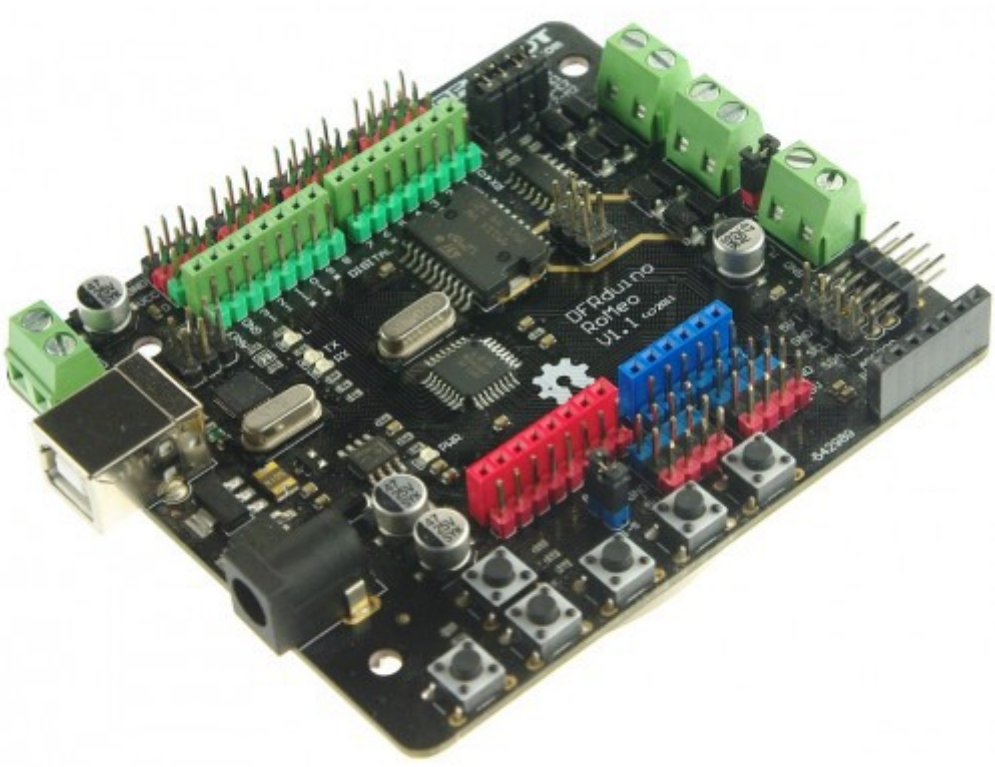
CARTE ROMEO

VERSION 1.1



Date de dernière mise à jour : 20/05/2014

Table des matières

1 - Introduction

La carte Romeo (disponible à l'adresse suivante: http://boutique.3sigma.fr/compatibles-arduino/11-carte-tout-en-un-romeo-atmega328-compatible-arduino.html) présente un rapport qualité-fonctionnalité / prix assez imbattable pour les applications de robotique ou de commande de moteur électrique.

Compatible Arduino Uno, elle possède un certain nombre de fonctionnalités uniques. Elle intègre en effet:

- 2 ponts en H pour le pilotage de 2 moteurs à courant continu jusqu'à 2A en continu pour chaque moteur: il est possible de piloter des moteurs assez puissant sans rajouter de shield ou de module externe
- un connecteur séparé pour l'alimentation des ponts en H: ceci permet de bien séparer l'alimentation du micro-contrôleur de l'alimentation des moteurs
- un connecteur séparé pour l'alimentation des servomoteurs: permet de fournir le courant nécessaire pour des servomoteurs puissants
- un connecteur pour modules APC220 RF et DF-Bluetooth: ajoute des possibilités de communication sans fil sans rajouter de shield ou de module externe
- différents type de broches (mâle, femelle et groupe Vcc / Masse / Signal), par ailleurs repérées par un code de couleur, pour chacune des entrées-sorties du micro-contrôleur: la connexion avec les capteurs ou actionneurs est beaucoup plus facile et ne nécessite par de câblage supplémentaire

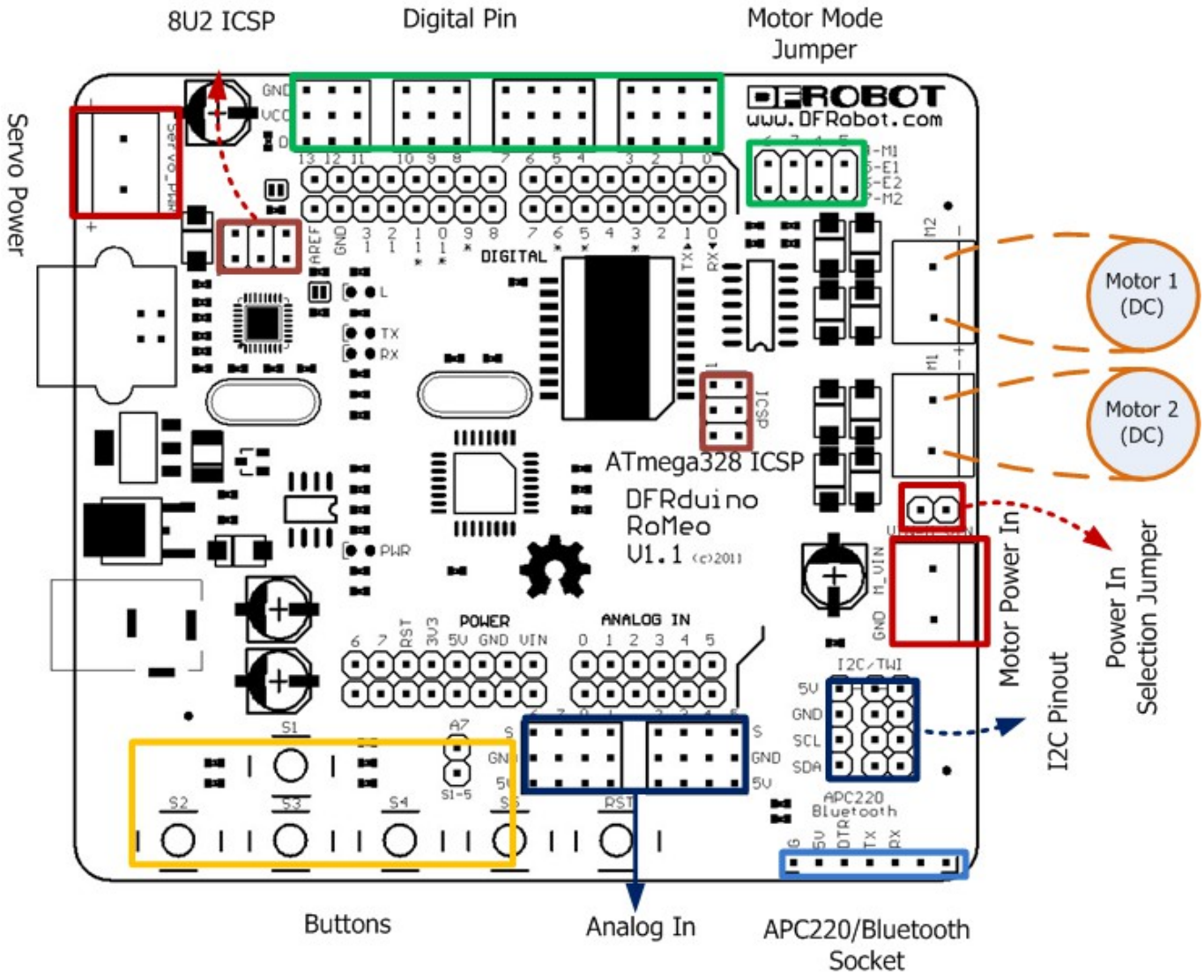
Voici un résumé de ses caractéristiques:

- Micro-contrôleur: Atmega328
- Bootloader: Arduino Uno
- Compatible avec le brochage de l'Arduino Uno
- 14 entrées-sorties digitales
- 6 voies PWM (broches 3, 5, 6, 9, 10 et 11)
- 8 entrées analogiques 10-bits
- Support de AREF
- 5 boutons poussoirs
- Interface USB
- Alimentation continue: USB ou externe 7V~12V DC
- Sortie tension continue: 5 V / 3.3 V DC et puissance
- Commutation d'alimentation automatique
- Connecteur ICSP pour programmation directe
- Interface série TTL
- Connecteurs mâle et femelle
- Connecteur intégré pour modules APC220 RF et DF-Bluetooth

- 3 jeux de broches (dont 2 à 90°) pour interface I2C
- Deux voies de pilotage de moteur à courant continu, 2 A maximum par voie
- PCB plaqué or
- Taille: 90 x 80 x 14 mm
- Poids: 60 g

2 - Détails

Voici un schéma du brochage de la carte Romeo:



Cette figure montre les lignes d'entrée-sortie et les connecteurs de la carte:

- Un bornier d'alimentation régulée (6V à 12V) pour les moteurs (« Moteur Power In », à droite en rouge)
- Un cavalier de sélection du mode d'alimentation (« Power In Selection Jumper », à droite en rouge)
- Une bornier d'alimentation non régulée (vous devez connecter une alimentation régulée entre 4V et 7.2V) pour les servomoteurs (« Servo Power », en haut à gauche en rouge)
- Une interface série de communication sans fil pour module APC220/Bluetooth (« APC220/Bluetooth Socket », en bas à droite en bleu ciel)
- 2 borniers pour moteurs à courant continu, 2A max par voie (« Motor 1 (DC) » et « Motor 2 (DC) », à droite en orange)
- 3 ports I2C/TWI (SDA, SCL, 5V, GND: « I2C Pinout » à droite en bleu)
- 8 entrées analogiques (« Analog In », en bas en bleu)
- 13 entrées-sorties digitales (« Digital Pin », en haut en vert). Les voies 4, 5, 6 et 7 peuvent être utilisées pour le contrôle de moteur électrique
- Des cavaliers de sélection du mode de commande des moteurs (« Motor Mode Jumper », en haut à droite en vert)
- 5 boutons (« Buttons », en bas en jaune) et un bouton Reset
- Un connecteur ICSP pour la programmation directe de l'ATmega328 (« Atmega328 ICSP », au milieu à droite en rouge)
- Un connecteur ICSP pour la programmation directe du micro-contrôleur d'interface USB/série («8U2 ICSP », au milieu à droite en rouge)

3 - Mise en œuvre

3.1 - Alimentation

Vous pouvez brancher sur la carte Romeo jusqu'à 3 alimentations:

- une alimentation pour les moteurs à courant continu
- une alimentation pour les servomoteurs
- une alimentation pour le micro-contrôleur

Lorsque vous utilisez la carte Romeo pour commander des moteurs (à courant continu ou servomoteurs), il est fortement recommandé de brancher leur alimentation spécifique (voir ci-dessous) car le régulateur de tension de l'alimentation du micro-contrôleur ne fournit pas plus de 500 mA, ce qui n'est en général pas suffisant pour alimenter des moteurs.

3.1.1 - Alimentation des moteurs

ATTENTION !
Il est impératif de faire très attention aux connexions de l'alimentation de la carte Romeo car celle-ci n'est pas protégée contre les inversions de polarité. Une erreur de connexion sur les bornes d'alimentation risque d'entrainer la destruction du sous-ensemble de gestion d'alimentation de la carte et de rendre celle-ci inutilisable.

Il est nécessaire d'avoir une alimentation séparée pour les moteurs à courant continu car ceux-ci consomment la plupart du temps plus de courant que ne peut fournir le régulateur de tension de l'alimentation du micro-contrôleur (connecteur jack ou USB).

Prenez votre temps pour réaliser les connexions d'alimentation et vérifiez plutôt deux fois qu'une avant de mettre la carte sous tension.

La tension positive doit être branchée sur la borne M_VIN du bornier « Motor Power In » (en rouge à droite sur le schéma du paragraphe 2). La masse doit être branchée sur la borne GND.

3.1.2 - Sélecteur d'alimentation

Un sélecteur d'alimentation « Power In Selection Jumper » (en rouge à droite sur le schéma du paragraphe 2) permet de spécifier l'alimentation utilisée pour alimenter le micro-contrôleur de la carte:

- jumper en place (VIN=M_VIN): le micro-contrôleur est alimenté par l'alimentation des moteurs. Dans cette configuration, il n'est pas nécessaire d'avoir une autre alimentation pour le micro-contrôleur (par connecteur jack ou par connexion USB avec l'ordinateur, comme sur les Arduino classiques). Cependant, ces différentes alimentations peuvent malgré tout cohabiter.
  L'intérêt de ce mode d'alimentation se trouve dans l'utilisation de la carte embarquée dans un robot mobile (comme Geeros, http://www.geeros.com, par exemple): l'alimentation des moteurs suffit alors pour tout alimenter.
- jumper enlevé: le micro-contrôleur **n'est pas** alimenté par l'alimentation des moteurs. Dans cette configuration, il **est nécessaire** d'avoir une autre alimentation pour le micro-contrôleur (par connecteur jack ou par connexion USB avec l'ordinateur, comme sur les Arduino classiques). Ce mode d'alimentation peut être intéressant si vous ne souhaitez pas démarrer le programme du micro-contrôleur dès la mise sous tension des moteurs (en effet, le programme Arduino démarre dès que le micro-contrôleur est alimenté).

3.1.3 - Alimentation des servomoteurs

Il est souvent nécessaire d'avoir une alimentation séparée pour les servomoteurs car si ceux-ci sont assez puissant, ils consomment plus de courant que ne peut fournir le régulateur de tension de l'alimentation du micro-contrôleur (connecteur jack ou USB).

Prenez votre temps pour réaliser les connexions d'alimentation et vérifiez plutôt deux fois qu'une avant de mettre la carte sous tension.

La tension positive doit être branchée sur la borne + du bornier « Servo Power » (en rouge à droite sur le schéma du paragraphe 2). La masse doit être branchée sur la borne -.

ATTENTION !
Cette alimentation est reliée à toutes les broches digitales. Il est très fortement recommandé de brancher une alimentation régulée 5V pour avoir un niveau de tension classique et éviter de détruire les éventuels capteurs qui seraient également branchés sur ces lignes digitales.

La sélection d'alimentation est automatique:

- si une alimentation est branchée sur le bornier « Servo Power », elle est utilisée pour alimenter les broches digitales
- si aucune alimentation n'est branchée sur le bornier « Servo Power », c'est l'alimentation 5V du micro-contrôleur qui est utilisée pour alimenter les broches digitales

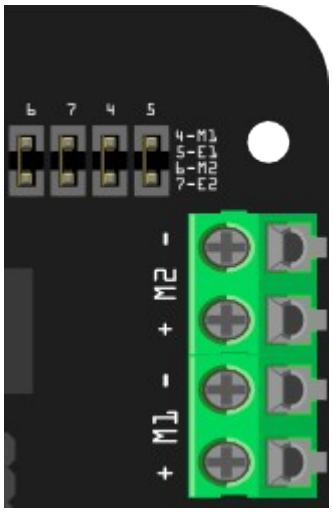
IMPORTANT !
Si vous n'avez pas besoin de puissance sur les lignes digitales, il est recommandé de ne pas brancher d'alimentation sur le bornier « Servo Power ».

3.2 - Contrôle de moteurs électriques

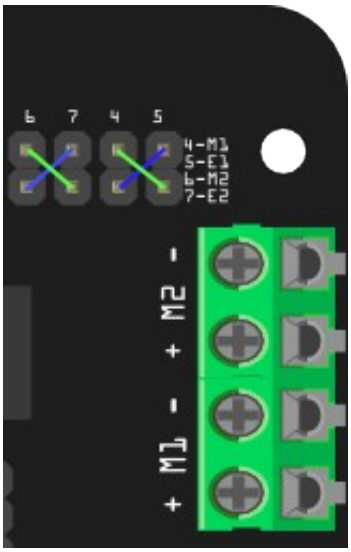
3.2.1 - Cavaliers de sélection du mode de contrôle

Le groupe « Motor Mode Jumper » (en vert en haut à droite sur le schéma du paragraphe 2) permet de sélectionner le mode de contrôle moteur:

- mode par défaut: contrôle des moteur en PWM



- mode PLL

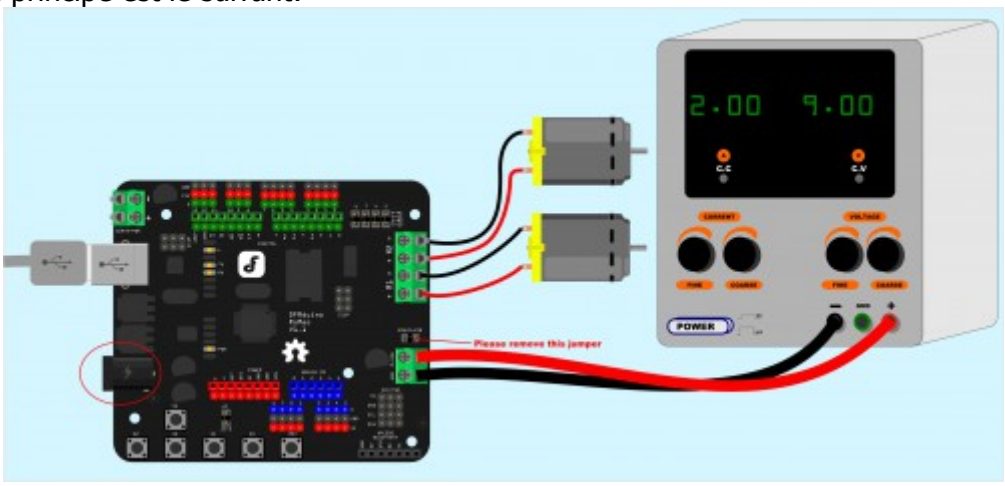


Il faut dans ce cas remplacer les cavaliers par des fils un peu plus longs

- sans aucune connexion: le contrôle de moteurs électriques est désactivé. Les broches 4, 5, 6 et 7 peuvent alors être utilisées en entrées-sorties digitales classiques

3.2.2 - Connexions matérielles pour piloter deux moteurs électriques

Le schéma de principe est le suivant:



3.2.3 - Contrôle en PWM

L'allocation des broches pour le mode de contrôle en PWM est la suivante:

Broche	Fonction
Digital 4	Contrôle de direction du moteur 1
Digital 5	Commande PWM du moteur 1
Digital 6	Commande PWM du moteur 2
Digital 7	Contrôle de direction du moteur 2

Voici un exemple de code:

```
// Contrôle PWM simple
int E1 = 5;     // Contrôle vitesse moteur 1
int E2 = 6;     // Contrôle vitesse moteur 2
int M1 = 4;    // Contrôle direction moteur 1
int M2 = 7;    // Contrôle direction moteur 2

void stop(void)                    //Stop
{
  digitalWrite(E1,LOW);
  digitalWrite(E2,LOW);
}
void advance(char a,char b)          // En avant
{
  analogWrite (E1,a);      // Contrôle de vitesse en PWM
  digitalWrite(M1,HIGH);
  analogWrite (E2,b);
  digitalWrite(M2,HIGH);
}
void back_off (char a,char b)          // En arrière
{
  analogWrite (E1,a);
  digitalWrite(M1,LOW);
  analogWrite (E2,b);
  digitalWrite(M2,LOW);
}
void turn_L (char a,char b)             // Tourne à gauche
{
  analogWrite (E1,a);
  digitalWrite(M1,LOW);
  analogWrite (E2,b);
  digitalWrite(M2,HIGH);
}
void turn_R (char a,char b)             // Tourne à droite
{
  analogWrite (E1,a);
  digitalWrite(M1,HIGH);
  analogWrite (E2,b);
  digitalWrite(M2,LOW);
}
void setup(void)
{
  int i;
  for(i=4;i<=7;i++)
    pinMode(i, OUTPUT);
  Serial.begin(19200);      // Définit vitesse de transmission série
  Serial.println("Exécution du contrôle par clavier");
}
void loop(void)
{
  if(Serial.available()){
    char val = Serial.read();
    if(val != -1)
    {
      switch(val)
      {
      case 'w':// En avant
        advance (255,255);   // en avant vitesse max
        break;
```

```
      case 's':// En arrière
        back_off (255,255);   // en arrière vitesse max
        break;
      case 'a':// Tourne à gauche
        turn_L (100,100);
        break;
      case 'd':// Tourne à droite
        turn_R (100,100);
        break;
      case 'z':
        Serial.println("Bonjour !");
        break;
      case 'x':
        stop();
        break;
      }
    }
    else stop();
  }
}
```

3.2.4 - Contrôle en PLL

L'allocation des broches pour le mode de contrôle en PLL est la suivante:

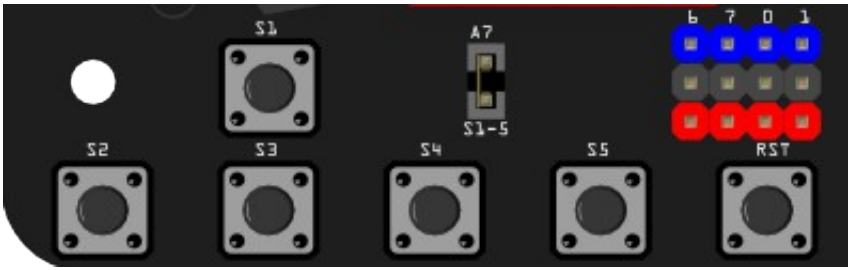
Broche	Fonction
Digital 4	Commande d'activation du moteur 1
Digital 5	Contrôle de direction du moteur 1
Digital 6	Contrôle de direction du moteur 2
Digital 7	Commande d'activation du moteur 2

Voici un exemple de code:

```
// Contrôle PLL simple

int E1 = 4;     // Contrôle vitesse moteur 1
int E2 = 7;     // Contrôle vitesse moteur 2
int M1 = 5;    // Contrôle direction moteur 1
int M2 = 6;    // Contrôle direction moteur 2

// Quand m1p/m2p = 127, cela arrête le moteur
// Quand m1p/m2p = 255, la vitesse est maximale dans une direction
// Quand m1p/m2p = 0, la vitesse est maximale dans la direction opposée

void DriveMotorP(byte m1p, byte m2p)
{

  digitalWrite(E1, HIGH);
  analogWrite(M1, (m1p));

  digitalWrite(E2, HIGH);
  analogWrite(M2, (m2p));

}

void setup(void)
{
  int i;
  for(i=6;i<=9;i++)
    pinMode(i, OUTPUT);
  Serial.begin(19200);      // Définit vitesse de transmission série
}
void loop(void)
{
  if(Serial.available()){
    char val = Serial.read();
    if(val!=-1)
    {
      switch(val)
      {
      case 'w':// En avant
        DriveMotorP(0xff,0xff); // Vitesse max
        break;
      case 'x':// En arrière
        DriveMotorP(0x00,0x00); // Vitesse max
        break;
      case 's':// Arrêt
        DriveMotorP(0x7f,0x7f);
        break;

      }
    }
  }
}
```

3.3 - Utilisation des boutons

En plus du bouton Reset, la carte Romeo possède 5 boutons:



Pour que ces boutons soient fonctionnels, ils doivent être liés à la voie d'entrée analogique n°7, ce qui est le cas lorsque le cavalier A7 (entre le bouton S1 et le groupe d'entrées analogiques sur la figure ci-dessus) est en place.

Voici un exemple de code:

```
130 char msgs[5][25] = {
131   "Bouton droit OK ",
132   "Bouton haut OK    ",
133   "Bouton bas OK  ",
134   "Bouton gauche OK  ",
135   "Bouton de selection OK" };
136 char start_msg[25] = {
137   "Demarrage de la boucle "};
138 int  adc_key_val[5] ={
139   30, 150, 360, 535, 760 };
140 int NUM_KEYS = 5;
141 int adc_key_in;
142 int key=-1;
143 int oldkey=-1;
144 void setup() {
145   pinMode(13, OUTPUT);  // Utilisation de la led de debug pour afficher une info
146   Serial.begin(9600);
147
148   /* Affichage des opération précédentes */
149   Serial.println(start_msg);
150 }
151
152 void loop()
153 {
154   adc_key_in = analogRead(7);    // lecture sur la voie analogique 7
155   digitalWrite(13, HIGH);
156   /* Lecture du bouton */
157   key = get_key(adc_key_in);    // conversion en bouton
158   if (key != oldkey) {   // si l'appui sur un bouton a été détecté
159     delay(50);      // attente pour stabilisation ("debounce")
160     adc_key_in = analogRead(7);    // lecture sur la voie analogique 7
161     key = get_key(adc_key_in);    // conversion en bouton
162     if (key != oldkey) {
163       oldkey = key;
```

```
      if (key >=0){
        Serial.println(adc_key_in, DEC);
        Serial.println(msgs[key]);
      }
    }
  }
  digitalWrite(13, LOW);
}
// Conversion de la valeur d'ADC en numéro de bouton
int get_key(unsigned int input)
{
  int k;
  for (k = 0; k < NUM_KEYS; k++)
  {
    if (input < adc_key_val[k])
    {
      return k;
    }
  }
  if (k >= NUM_KEYS)
    k = -1;     // Aucune bouton valide appuyé
  return k;
}
```

3.4 - Communication sans fil

Le connecteur « APC220/Bluetooth socket » permet de connecter un module APC220 RF ou un module DF-Bluetooth pour avoir une communication série sans fil entre la carte Romeo et l'ordinateur ou entre la carte Romeo et un autre dispositif sans fil utilisant le même type de communication.

IMPORTANT !
La carte Romeo ne possédant qu'une seule ligne série, il est impératif de retirer le module radio pour programmer la carte via sa liaison USB

4 - Installation de l'IDE Arduino

La carte Romeo se programme traditionnellement avec l'IDE Arduino, que vous pouvez télécharger à l'adresse suivante: http://arduino.cc/en/Main/Software. Pour l'installer, il suffit de décompresser l'archive téléchargée dans le répertoire de votre choix. Notez bien ce répertoire car vous aurez besoin de le retrouver si vous devez installer des librairies additionnelles.

5 - Connexion à l'ordinateur

5.1 - Windows

Lors de la toute première connexion, si vous n'avez jamais installé de carte compatible Romeo, il s'affiche une fenêtre indiquant que Windows doit installer le pilote de votre Arduino Uno (la carte Romeo est en effet considérée comme un Arduino Uno).

Sélectionner « Rechercher et installer le pilote logiciel », ce qui est l'action recommandée par Windows.

Si on vous demande d'insérer un disque, ne le faites pas puisque vous n'en avez pas. Il faut sélectionner une autre option, qui consiste à installer le pilote manuellement. Celui-ci se trouve dans le répertoire dans lequel vous avez installé l'IDE Arduino, sous dossier « drivers » (laissez cochée la case « Inclure les sous-dossiers »).

Windows peut alors trouver le pilote et l'installer.

6 - Utilisation de l'IDE Arduino

Vous pouvez alors utiliser l'IDE Arduino pour programmer la carte.

Lors de son tout premier lancement après la connexion de votre carte Romeo et l'installation de son pilote, vous devez choisir la carte cible et le port de communication série utilisé:

- Allez dans Outils → Type de carte et choisissez « Arduino Uno »
- Puis allez dans Outils → Port série et choisissez le dernier port COM de la liste (a priori celui-ci doit correspondre à celui que vous venez d'installer)