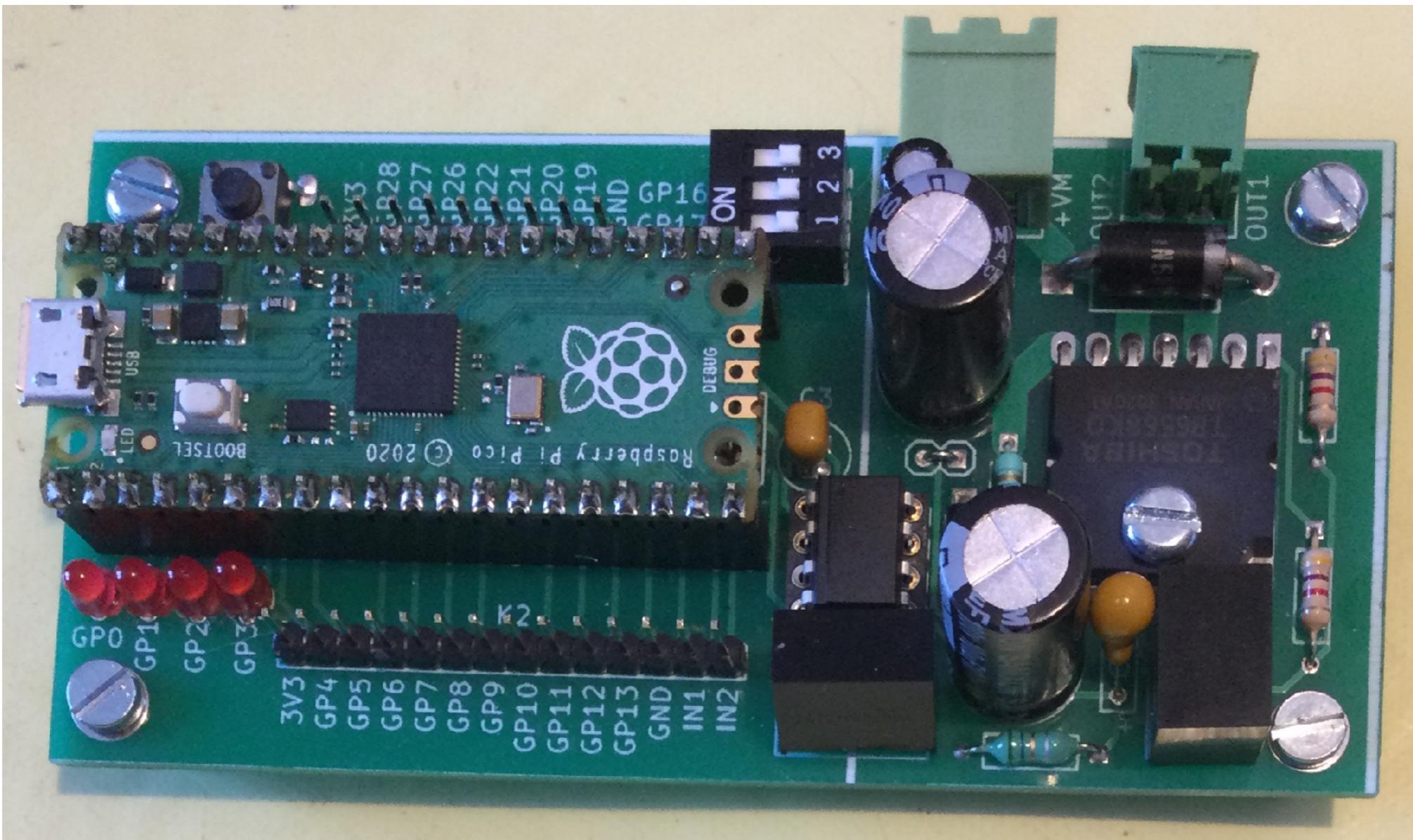


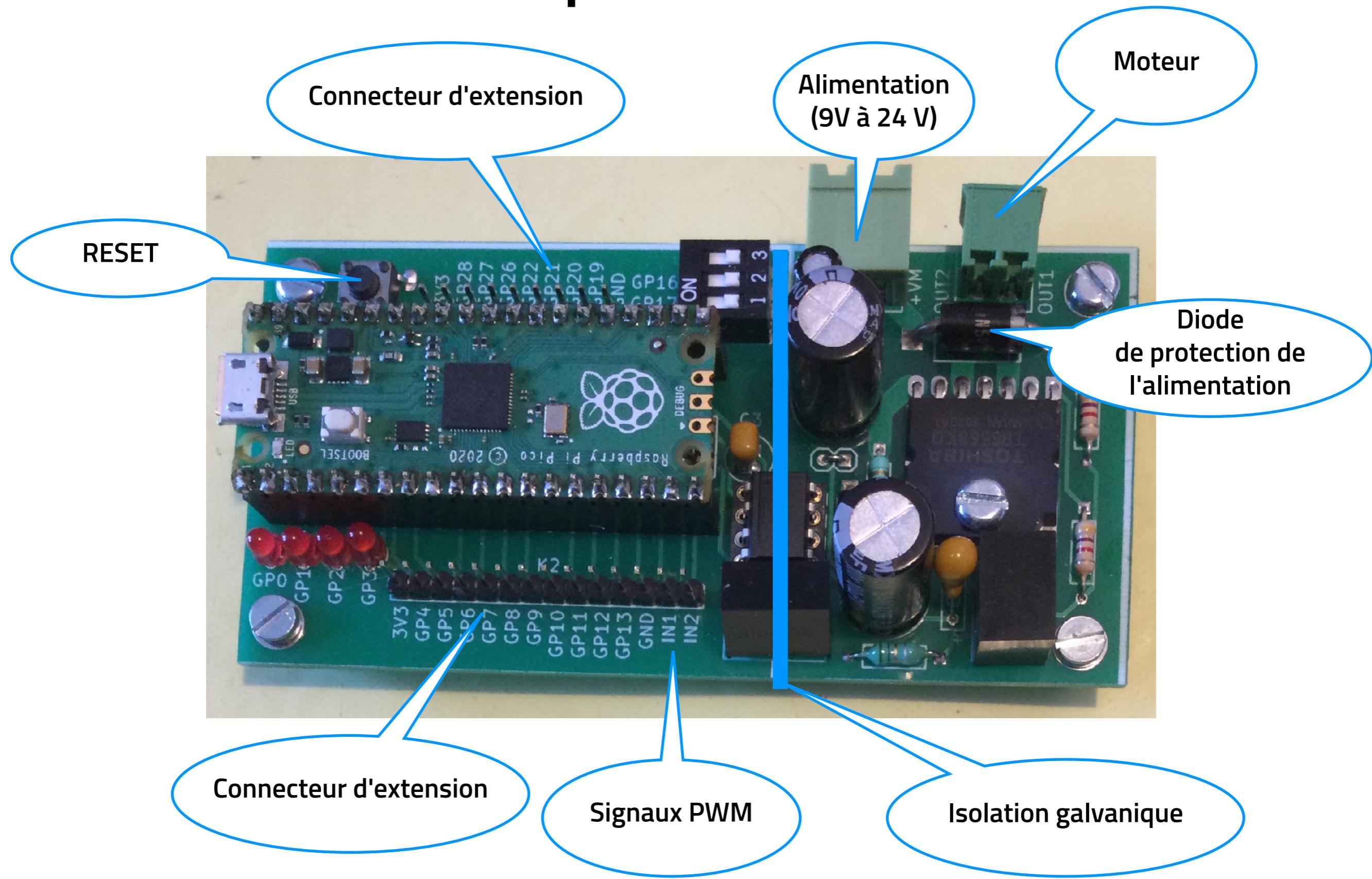
# Pont en H avec Raspberry Pi Pico



<https://github.com/pierremolinaro/cartes-micro-controleurs-centrale-nantes/tree/master/pont-en-h-ohr>

**Pierre Molinaro**  
**18 juin 2023**

# Description de la carte



# Commande du pont en H

Le pont en H intégré est commandé par GP14 et GP15 :

- GP14 : sortie commande moteur IN1 ;
- GP15 : sortie commande moteur IN2.

La commande est transmise via un opto-coupleur (MCT62), sans inversion (GP14 au niveau bas -> IN1 au niveau bas, GP14 au niveau bas -> IN1 au niveau haut, idem pour GP15 et IN2).

Le croquis d'exemple **croquis-test.ino** contient un exemple de programmation d'une PWM sur GP14 et GP15 (fichiers **pwm.cpp** et **pwm.h**).

Quatre fonctions sont définies pour contrôler facilement les deux sorties :

```
void configurerPWM (const uint32_t inFrequencePWM,  
                     const uint32_t inNombreNiveauxPWM) ;  
  
void marcheAvant (const uint32_t inRapportCyclique) ; // Entre 0 et inNombreNiveauxPWM  
  
void marcheArriere (const uint32_t inRapportCyclique) ; // Entre 0 et inNombreNiveauxPWM  
  
void arret (void) ;
```

# Commande du pont en H : fonction configurerPWM

```
void configurerPWM (const uint32_t inFrequencePWM,  
                    const uint32_t inNombreNiveauxPWM) ;
```

Cette fonction est à appeler une fois pour configurer la PWM (typiquement dans la fonction setup). Elle place les sorties GP14 et GP15 au niveau bas (donc le moteur est arrêté).

**inFrequencePWM** est la fréquence de la PWM.

**inNombreNiveauxPWM** est le nombre de niveaux.

Exemple : configurerPWM (5000, 256) ;

# Commande du pont en H : fonction marcheAvant

```
void marcheAvant (const uint32_t inRapportCyclique) ; // Entre 0 et inNombreNiveauxPWM
```

Cette fonction permet de commander le moteur dans le sens dit « marche avant » : GP15 est au niveau bas, GP14 est une PWM contrôlée par `inRapportCyclique` :

- `inRapportCyclique` = 0, GP14 est au niveau bas (donc le moteur est arrêté) ;
- `inRapportCyclique` = `inNombreNiveauxPWM`, GP14 est au niveau haut (donc le moteur est au maximum) ;
- `inRapportCyclique` = `inNombreNiveauxPWM` / 4, GP14 est une PWM donc le rapport cyclique est 25 %.

# Commande du pont en H : fonction marcheArriere

```
void marcheArriere (const uint32_t inRapportCyclique) ; // Entre 0 et inNombreNiveauxPWM
```

Cette fonction permet de commander le moteur dans le sens dit « marche arrière » : GP14 est au niveau bas, GP15 est une PWM contrôlée par `inRapportCyclique` :

- `inRapportCyclique` = 0, GP15 est au niveau bas (donc le moteur est arrêté) ;
- `inRapportCyclique` = `inNombreNiveauxPWM`, GP15 est au niveau haut (donc le moteur est au maximum) ;
- `inRapportCyclique` = `inNombreNiveauxPWM` / 4, GP15 est une PWM donc le rapport cyclique est 25 %.

# **Commande du pont en H : fonction arret**

```
void arret (void) ;
```

Cette fonction arrête le moteur : GP14 et GP15 sont placés au niveau bas.

# Le croquis croquis-test.ino

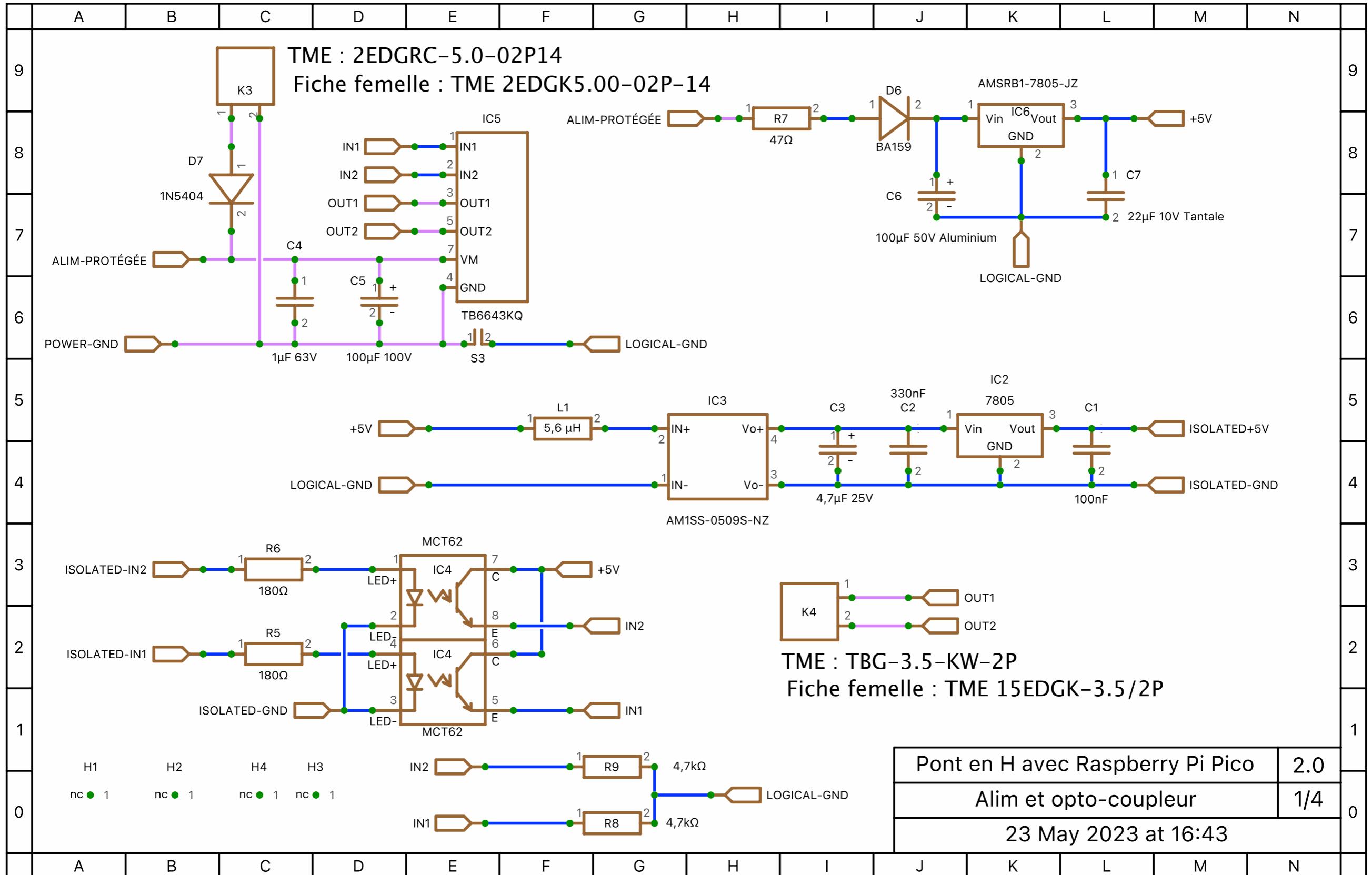
Ce croquis allume les quatre leds cycliquement.

Les interrupteurs DIL2 (GP17) et DIL3 (GP16, marqué par erreur GP18 sur la sérigraphie) contrôlent le moteur :

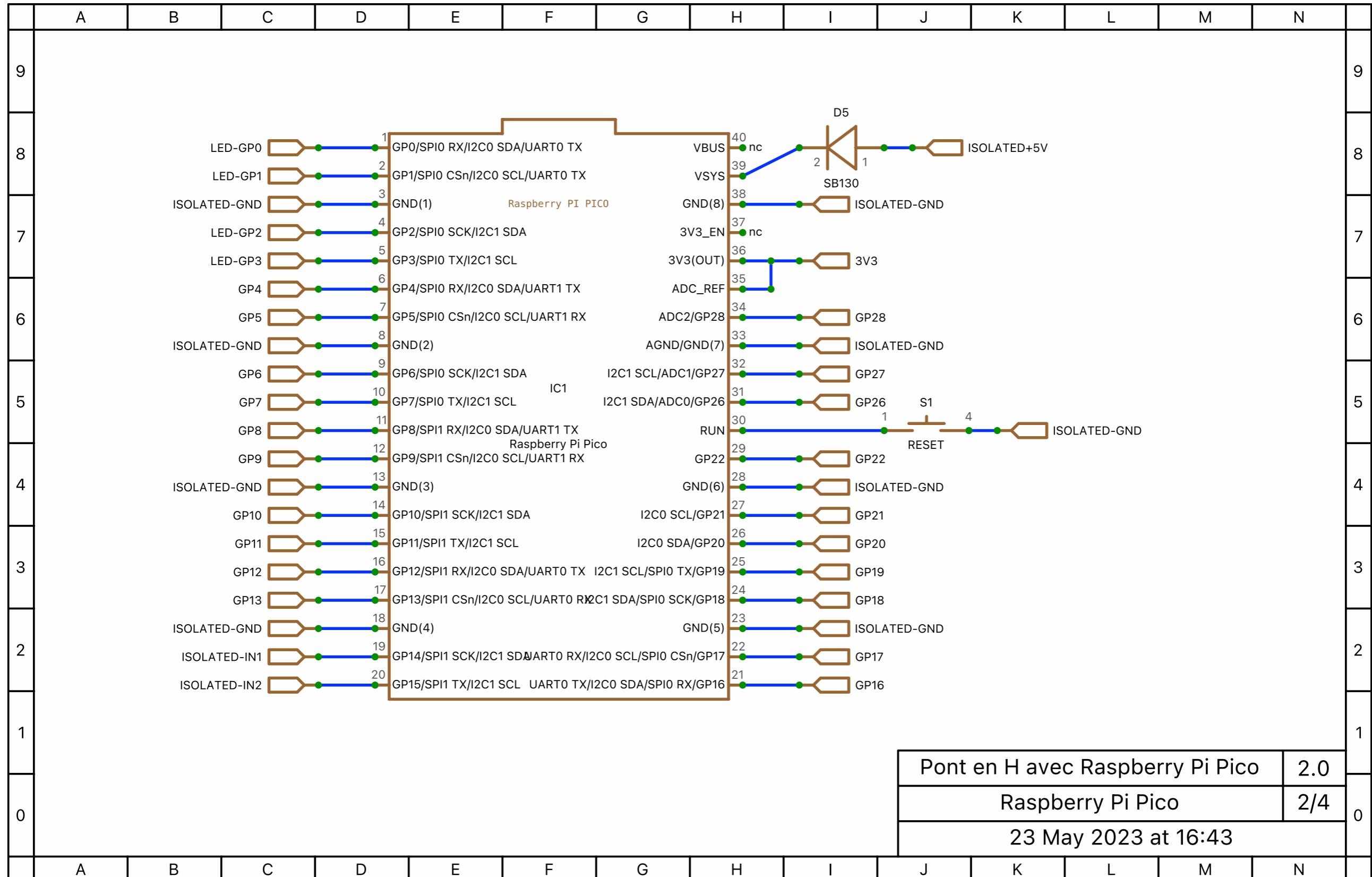
- OFF et OFF : arrêt ;
- ON et OFF : marche arrière ;
- OFF et ON : marche avant ;
- ON et ON : arrêt.

Pour les croquis destinés aux TP, il y a libre choix d'utiliser comme on veut les leds, les interrupteurs DIL et et les ports sur le connecteurs d'extension.

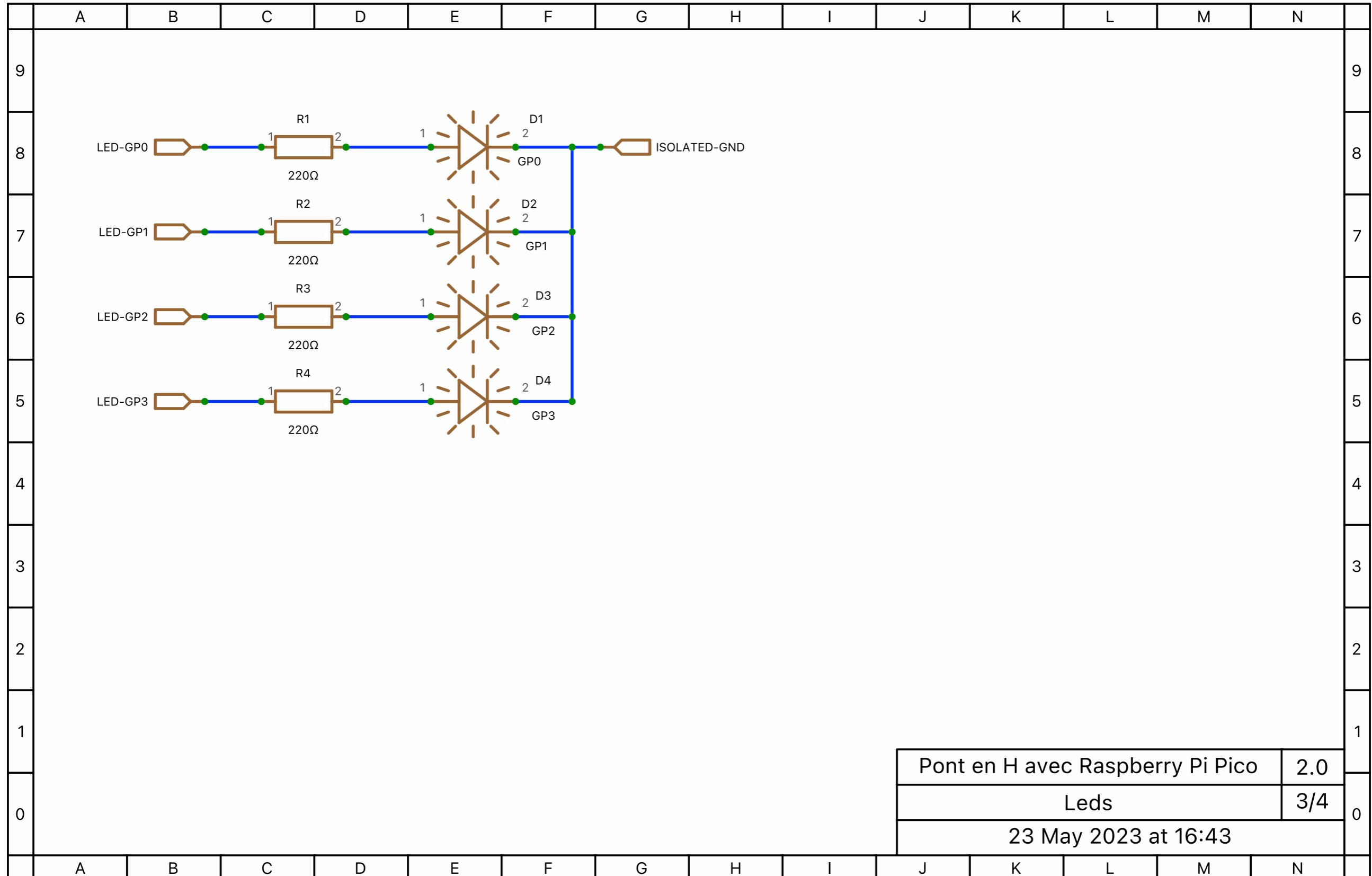
# Plan de la carte (1/4)



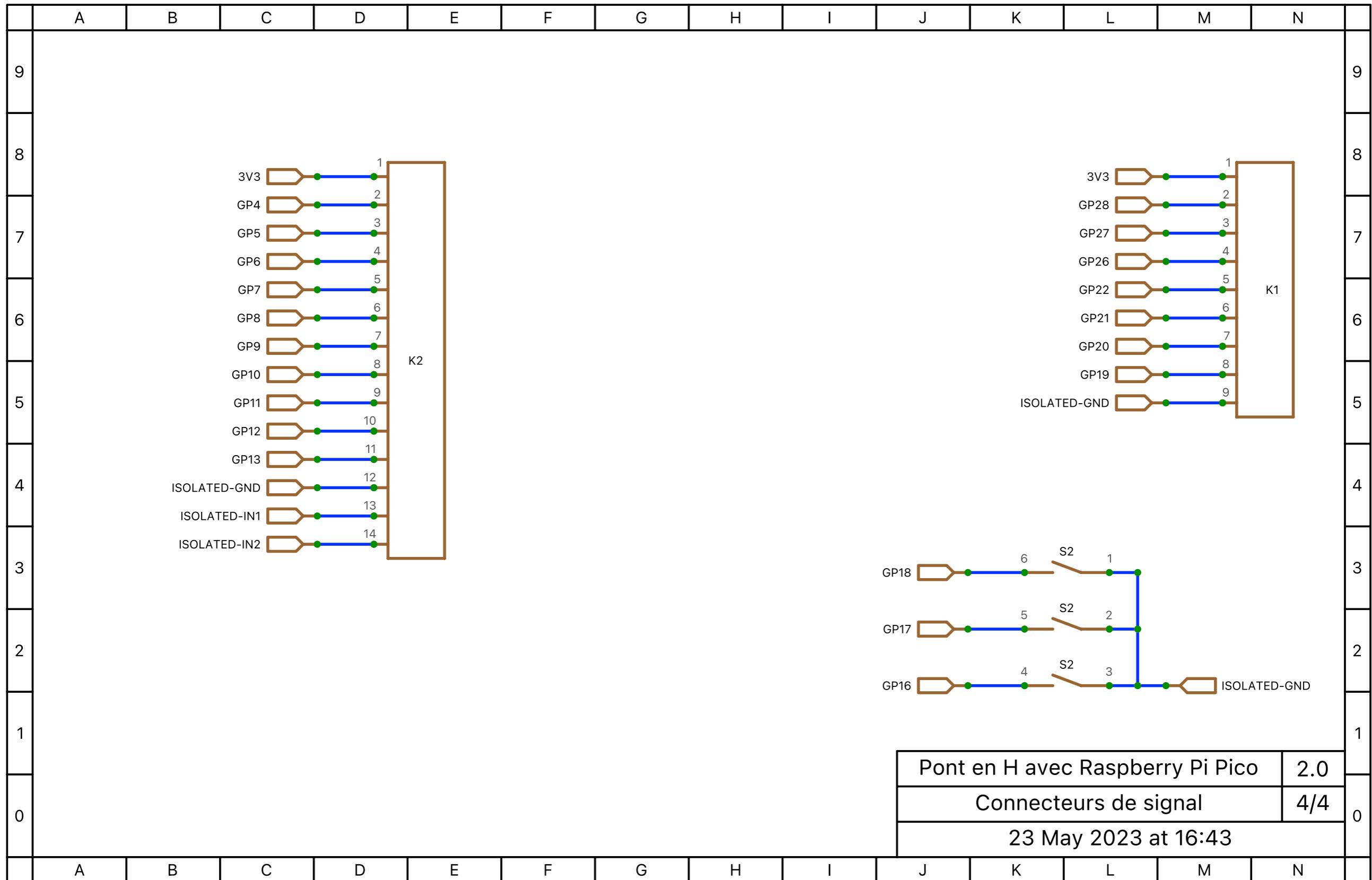
# Plan de la carte (2/4)



# Plan de la carte (3/4)



# Plan de la carte (4/4)



# Circuit imprimé

