REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE SUPERIEURE EN SCIENCES APPLIQUEES
-T L E M C E N-

وزارة التعليم العالي والبحث العلمي

المدرسة العليا في العلوم التطبيقية
-تلمسان-

École Supérieure En Sciences Appliquées de Tlemcen

Département de la formation du Second Cycle

Filière: Électrotechnique

Polycopié des Travaux Pratiques

Systèmes embarqués et temps réel (Raspberry Pi)

Élaboré Par
Dr ABDELLAOUI Ghouti
Dr MEGNAFI Hicham

Table des matières

Table des figures

Préface

Ce polycopié, couvre un ensemble des travaux pratiques dédiés aux élevés ingénieurs de la quatrième année électrotechnique. Il résume les principes de base pour la conception et la réalisation des systèmes embarqués à base d'un système d'exploitation, la raison pour la quelle la carte Raspberry Pi version 3 modèle B a été choisie comme une plate forme de développement sachant qu'elle fonctionne avec le système d'exploitation Linux (Raspbian).

L'objectif pédagogique de ces travaux pratiques, sert à inculquer aux étudiants les notions de base de la programmation des systèmes embarqués, de sorte qu'ils développent en utilisant le langage C leur propre bibliothèque pour la manipulation des entrées sorties de la Raspberry Pi. L'implémentation des différentes fonctions de la bibliothèque, permet aux étudiants de manipuler correctement et librement les différents ports GPIOs de la Raspberry Pi et comprendre le système de gestion de fichiers **linux**.

Pour le bon déroulement de ces travaux pratiques, l'étudiant doit avoir un ensemble de prérequis de base en électronique numérique, architecture matérielle des systèmes à microprocesseur, les réseaux informatiques industriels et la programmation en langage C.

Les objectifs ciblés sont :

- Installer et configurer un systèmes d'exploitation sur une plate-forme embarqué,
- Développer des applications multi-tâches pour gérer un ensemble de capteurs,
- Concevoir des systèmes de contrôle et/ou de commande en temps réel,

Introduction générale

1 Introduction

L'évolution technologique a créé un nouveau monde dédié au traitement de l'information, où n'importe quel composant réel est caractérisé par deux ensembles (matériel et traitements). Ce dernier peut être vu comme un environnement global d'observation pour plusieurs systèmes dans les différents domaines comme le contrôle des équipements médicaux, l'assistance autonome, le contrôle et la sécurité du trafic, les systèmes avancés automobile, le contrôle du traitement, la conservation d'énergie, le contrôle de l'environnement, le contrôle des infrastructures critiques (l'énergie électrique, les système de communications par exemple : télé-présence, télé-médecine), les systèmes militaires, les systèmes de manufacturing, et toutes les structures intelligentes, etc.

La majorité de ces systèmes sont critiques et nécessitent un traitement de l'information à l'intérieur du système lui-même ce qui exige l'utilisation d'un système embarqué.

2 Système embarqué

Le développement technologique a donné naissance à un nouveaux type d'équipements qui occupent notre vie quotidienne, et que nous pouvons les trouvés partout :

— Radio/réveil

— Machine à café
— Télévision / télécommande
— Moyen de transport
— Téléphone portable
— ...

Ce genre d'équipements est caractérisé par la dotation des puces d'ordinateur qui sont constituées d'une unité de calcul (CPU), des mémoires et des programmes. Techniquement parlons, ce genre d'équipement est appelé *un système embarqué* :

Définition *Un système embarqué est défini comme un système électronique et informatique autonome, souvent temps réel, spécialisé dans une tâche bien précise. Ses ressources sont généralement limitées.*

Certains systèmes embarqués sont complexes,ils nécessitent ***un système d'exploitation*** *pour gérer tous les ressources matérielles.*

3 Système d'exploitation

Définition *En informatique un système d'exploitation est une interface logiciel qui permet de faire le lien entre :*

— *L'utilisateur,*
— *Les applications,*
— *Les composantes matérielles d'un ordinateur.*

3.1 Les composants d'un système d'exploitation

Un système d'exploitation est généralement composé de :

— Le noyau (kernel) :
  — Fonctions de base pour gérer la mémoire, les tâches exécutées, les fichiers, les entrées sorties principales, et les fonctionnalités de communication.

- L'interpréteur de commande (**shell**) :
  - Permet de communiquer avec le noyau via un langage de commande prédéfini.
  - Une version à ligne de commande alphanumérique coexiste souvent avec une version à interface graphique (GUI).
- Le système de fichiers (file system ou FS) :
  - Permet l'enregistrement des fichiers dans une structure de répertoires arborescente.

Durons les travaux pratiques qui suivent, nous allons utiliser le **shell** et **le système de fichiers** pour réaliser des commandes de contrôle d'un ensemble d'entrées sorties.

3.2 Système d'exploitation embarqué

Dans un système embarqué les applications et le système d'exploitation sont fusionnés de tel manière :

- Le code du SE et celui de l'application résident dans de la mémoire non volatile.
- La gestion des ressources est spécifique à celles présentes.
- Moins de surcharge de traitement (overhead).

4 Organisation des travaux pratiques

Dans ce polycopié des travaux pratiques, nous allons débuter par l'installation et la préparation du système d'exploitation Raspbian sur une Raspberry Pi version 3 modèle B, afin de donner la possibilité aux étudiants de développer leur propre bibliothèque en C. Cette dernière va contenir tous les fonctions essentielles pour la manipulation des ports GPIO de la Raspberry Pi, ceci sera au fur et à mesure de l'avancement des travaux pratiques, à partir des simples systèmes vers des systèmes plus complexes. Dans ce contexte les travaux pratiques serons organisés comme suites :

- TP N°1 : Prise en main d'un système embarqué Rapsberry PI.
- TP N°2 : Manipulation des Fichiers Entrées/Sorties (Raspbian) en C.
- TP N°3 : Commande d'un moteur à courant continu avec Raspberry Pi en C.

— TP N°4 : Régulation de la température ambiante à base d'un moteur DC.

— TP N°5 : Transmission de données via des interfaces hertziennes

TP N° 1

Prise en main d'un système embarqué Rapsberry PI

1.1 Introduction

Les ordinateurs embarqués dotés d'un système d'exploitation Linux sont massivement présents dans les technologies modernes (transports, multimédia, téléphonie mobile, appareils photos ...).

L'ordinateur Raspberry PI constitue un support d'apprentissage performant, très bon marché et disposant d'une forte communauté sur le net. Il possède des entrées/sorties puissantes permettant une connexion avec le monde physique par l'intermédiaire de capteurs et d'actionneurs.

1.2 Évaluation

L'évaluation portera sur :

- Un compte rendu détaillé explicitant les différentes opérations menées durant le TP. Pensez à prendre des copies d'écran pour illustrer ces opérations.
- La bonne installation du noyau Linux sur la carte Raspberry Pi et le test.

1.3 Description de l'ordinateur embarqué

1.3.1 Présentation

La carte Raspberry Pi représentée dans la figure 1.1 est un petit ordinateur qui fonctionne sous le système d'exploitation Linux, ce dernier est installé sur une carte SD destinée à des applications de l'informatique embarquée.

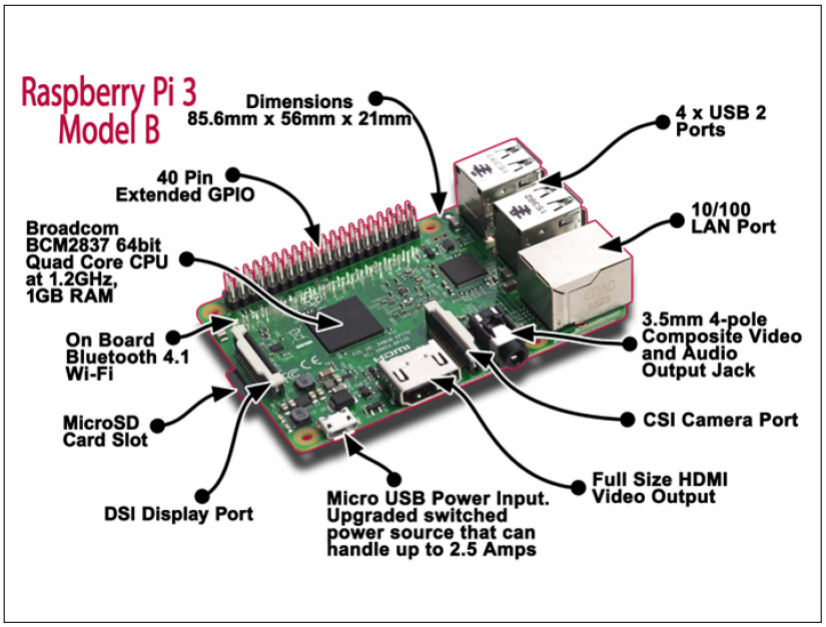


FIGURE 1.1: Carte mère Raspberry Pi 3 Type B

Carte mère Raspberry Pi 3 Type B :

- Processeur intégré Quad-core ARM Cortex-A53 1.2 GHz (Broadcom BCM2837).
- RAM : 1024 Mo.
- GPU Dual Core VideoCore IV Multimedia Co-Processor.
- Lecteur de cartes Micro SD.
- Ports : HDMI, 4x USB, RJ45, jack 3.5 mm, connecteurs pour APN et écran tactile.
- Wi-Fi b/g/n et Bluetooth 4.1.
- Support des distributions dédiées basées sur Linux et Windows 10.

1.3.2 Les connecteurs d'extension

Le connecteur d'extension de la carte Raspberry PI est utilisé pour raccorder des périphériques de communication (UART, I2C, SPI, ...) ou TOR (Tout Ou Rien).

Les broches peuvent avoir des fonctions différentes selon leurs configuration initiale : soit en tant que GPIO (Global Purpose Input Output), périphérique de communication ou sorties PWM (Pulse Width Modulation), comme le montre la figure 1.2 qui illustre les différentes configuration possible pour les broche GPIO.

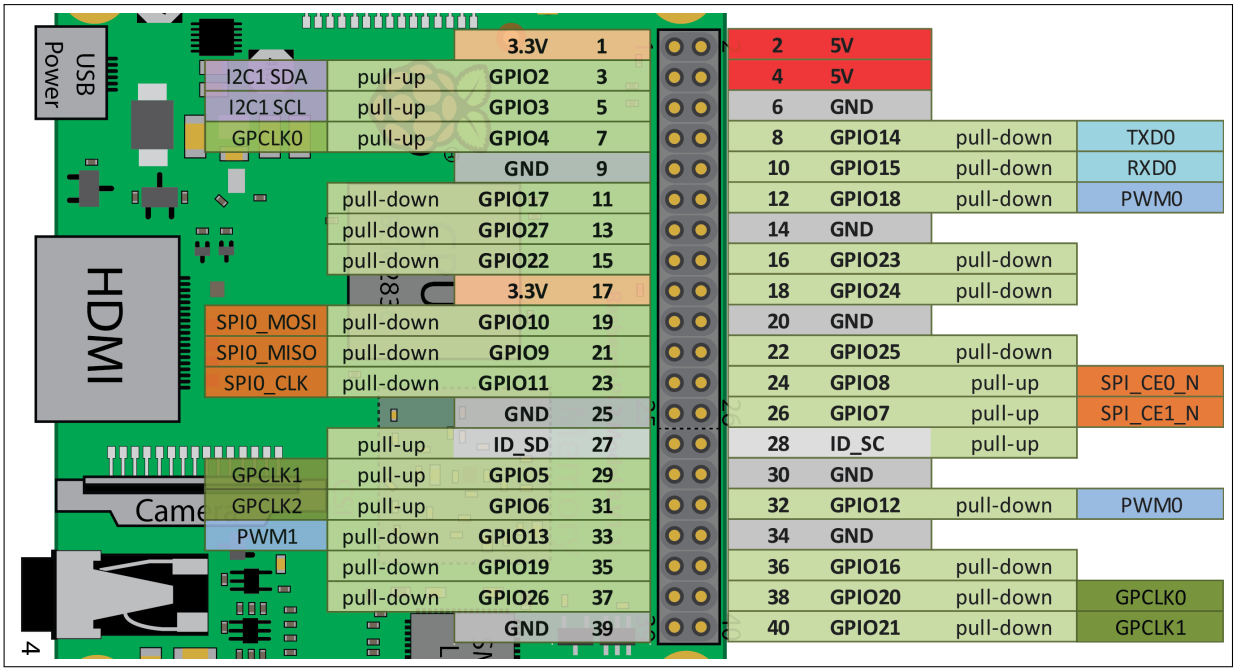


FIGURE 1.2: Les connecteurs d'extension GPIO

1.3.3 Le système d'exploitation Raspbian

Le Raspbian est un système d'exploitation libre basé sur la distribution GNU/Linux Debian, et optimisé pour le plus petit ordinateur du monde, la Raspberry Pi.

Raspbian ne fournit pas simplement un système d'exploitation basique, il est aussi livré avec plus de 35 000 paquets, c'est-à-dire des logiciels pré-compilés livrés dans un format optimisé, pour une installation facile sur votre carte Raspberry Pi via les gestionnaires de paquets.

La carte Raspberry Pi est une framboise merveilleuse, mais elle reste néanmoins dotée d'une puissance inférieure à celle d'un ordinateur moderne. Par conséquent, il est préférable d'installer un système optimisé pour la Raspberry.

1.4 Préparation de l'environnement de travail

Avant de brancher quoi que ce soit à votre Raspberry Pi, assurez-vous de disposer de l'ensemble des éléments listés ci-dessous. Puis, suivez les instructions d'installation.

- Une carte mémoire SD.
- Une distribution Raspbian, par exemple "2017-09-07-raspbian-stretch.img"
  http ://downloads.raspberrypi.org/raspbian_latest
- Le logiciel **Win32DiskManager** pour installer les fichier ".img" sur la carte SD.
- Un lecteur de carte SD externe.

1.4.1 Installation du fichier image sur la carte SD

Insérez votre carte SD dans le lecteur de votre ordinateur, et une fois celle-ci reconnue, lancez Win32DiskImager et cliquez sur l'icône représentant un dossier à droite du champ «Image File» comme le montre la figure 1.3.

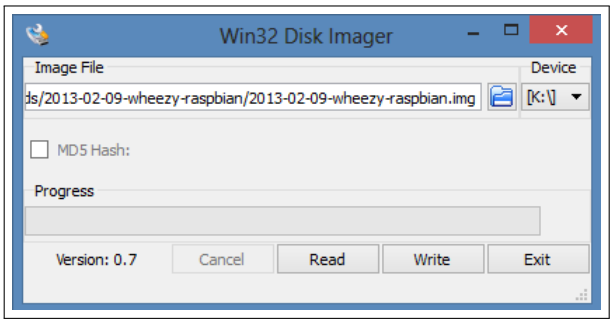


FIGURE 1.3: Interface d'utilisation du logiciel Win32DiskImager

Sélectionnez l'image adéquate à la distribution Raspbian, puis choisissez dans le device votre lecteur de carte mémoire, et à la fin cliquez sur Write pour lancer la gravure de l'image sur la carte SD.

1.4.2 Configuration du Raspbian

Maintenant que Raspbian est installé sur la carte SD, nous allons pouvoir configurer et modifier certains fichiers afin que la Raspberry Pi puisse fonctionner sans écran et sans clavier. Pour cela, nous allons commencer par activer SSH sur la Raspberry Pi.

Activation du protocole SSH

Par défaut les connexions SSH ne sont pas activés. La fondation Raspberry Pi a mis en place une solution simple et rapide pour activer le SSH. Il vous suffit de créer un fichier nommé SSH dans la partition boot (le fichier n'attend aucune extension).

Lors du premier démarrage de la carte Raspberry Pi, celle-ci vérifie si le fichier existe et activera le SSH en conséquence.

1.4.3 Configuration de l'adresse IP statique de la Raspberry Pi

Editez ou créez le fichier suivant : **/etc/network/interfaces**. et ajoutez les lignes suivantes :

```
auto eth0
iface eth0 inet static
address 192.168.1.2
netmask 255.255.255.0
network 192.168.1.0
broadcast 192.168.1.255
gateway 192.168.1.1
```

La carte SD est maintenant prête à être insérée dans la Raspberry Pi et démarrée sans écran et sans clavier. Vous n'avez plus qu'à vous y connecter en SSH à partir d'un ordinateur distant.

1.4.4 Connexion SSH sous Windows

Le logiciel PuTTY fonctionne sur les plateformes en 32bits de Windows et sur Unix. Il permet un accès au protocole SSH qui est ajouté au Telnet sous Windows, pour rendre la connexion plus fiable et les transferts plus sécurisés.

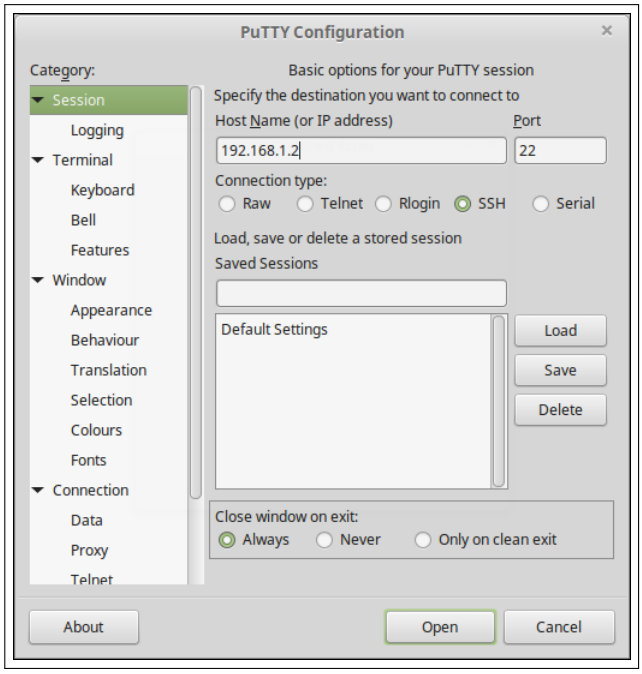


FIGURE 1.4: Interface d'utilisation du logiciel PuTTY

Après avoir téléchargé et installé PuTTY du site http ://www.putty.org/ (c'est un fichier unique appelé putty.exe), exécuté le programme.

Entrer l'adresse IP que vous avez configurée précédemment et cliquez sur "Open", comme le montre la figure 1.4. Cela vous donnera un avertissement (la première fois) et vous demandera alors l'utilisateur ("**pi**") et mot de passe ("**raspberry**"). La figure 1.5 montre la fenêtre SSH qui sera affichée et prête à être utilisée.

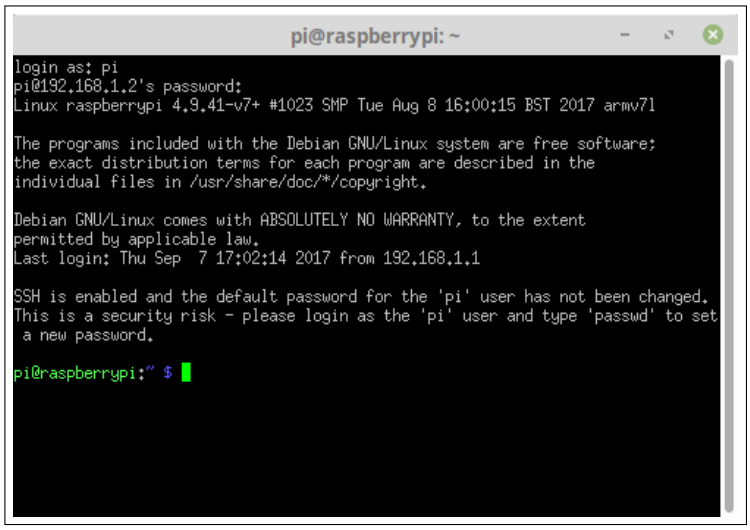


FIGURE 1.5: Connexion SSH à l'aide du logiciel PuTTY

1.5 Réalisation et test

Un des grands intérêts de la carte Raspberry Pi est qu'elle dispose des ports GPIO utilisables comme entrée ou sortie afin de lire des capteurs ou commandes des systèmes. Nous nous verrons ici comment contrôler les GPIO du Raspberry pi configurés en mode sortie, à travers un exemple concret où nous commanderons une LED.

1.5.1 Base d'expérimentation

Pour notre expérimentation, il nous faut :

- Une carte Raspberry Pi opérationnelle,
- Une LED,
- Une résistance de 220 Ω.

1.5.2 Schéma de câblage

La figure 1.6 illustre notre premier schéma de câblage pour tester le bon fonctionnement de la carte Raspberry Pi à l'aide d'une LED tels que :

- L'anode de la diode LED (patte longue) est raccordée à la sortie GPIO17 (broche 11).

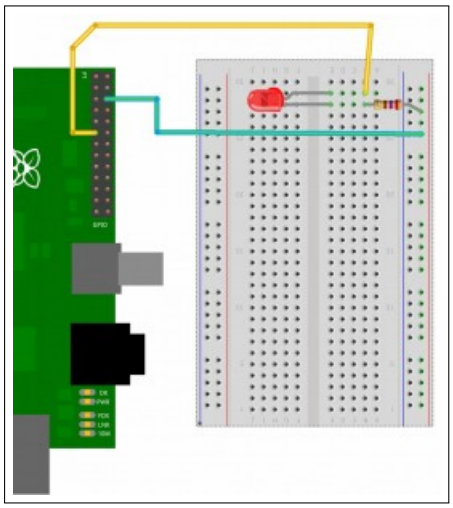


FIGURE 1.6: Schéma de câblage

— La cathode (patte courte) est reliée à la masse de la carte Raspberry Pi (Ground), par exemple la broche 6.

1.5.3 Cahier des charges

Tout programmeur qui se respecte doit disposer ou établir un cahier des charges qui définit les différentes opérations à réaliser. Dans notre cas, il sera très simple. Notre cahier des charges :

1. Allumer la LED.
2. Attendre 1 seconde.
3. Éteindre la LED.
4. Attendre 1 seconde.
5. Retour a 1 (on répète indéfiniment la boucle).

1.5.4 Implémentation du programme de commande

Dans ce premier TP et comme première étape, nous allons utiliser la ligne de commande pour tester la LED, pour ceci vous devez exécuter les commandes suivantes dans votre terminal tout en respectant l'ordre suivant :

1. Activer la GPIO17 : **echo "17" > /sys/class/gpio/export**

2. Configurer GPIO17 en sortie : **echo "out" > /sys/class/gpio/gpio17/direction**
3. Allumer LED : **echo "1" > /sys/class/gpio/gpio17/value**
4. Éteindre LED : **echo "0" > /sys/class/gpio/gpio17/value**

Deuxième étape nous allons regrouper tous les commandes ci-dessus dans un seul fichier système appelé **bash**

Pour saisir ce programme, utiliser l'éditeur de texte **nano** pour créer un fichier nommé *blink.sh*, en introduisant dans la console la ligne de commande suivante :

$nano blink.sh

Cette commande ouvre un fichier texte vide appelé blink.sh (pour la sauvegarde utiliser les touches clavier **Ctrl+o** et pour sortir **Ctrl+x**).

Le contenu du programme et de ses commentaires sont représentés ci-dessous.

```
echo "17" > /sys/class/gpio/export                    #assigne pin
echo "out" > /sys/class/gpio/gpio17/direction         #en sortie
while true;                                           #boucle infinie
do
echo "1" > /sys/class/gpio/gpio17/value               #allume LED
# echo "LED allumée"                                  #message IHM
sleep 1;                                              #attente 1 sec
echo "0" > /sys/class/gpio/gpio17/value               #éteint LED
# echo "LED éteinte"                                  #message IHM
sleep 1;                                              #attente 1 sec
done
```

1.6 Conclusion

Nous savons maintenant commander un GPIO pour allumer et éteindre une LED via les GPIO. Nous pouvons contrôler d'autres dispositifs qu'une LED, mais il convient de prendre en compte certaines limitations. En effet, les GPIO ne sont pas pensés pour alimenter des dispositif puissants. Une ou deux LED ne poseront pas de problème, mais il ne faut pas alimenter un moteur depuis le 3.3V. Si l'on souhaite commander une charge plus importante depuis un GPIO, il faudra utiliser un relais pour commander un circuit de grande puissance ou en courant alternatif. Dans tous les cas, il convient de ne pas consommer plus de **50mA** sur le **3.3V** et l'ensemble des GPIO en sortie.

TP N° 2

Manipulation des Fichiers Entrées/Sorties (Raspbian) en C

2.1 Introduction

Durant ce TP nous allons réaliser une bibliothèque en C qui regroupe un ensemble de fonctions permettant la manipulation des E/S GPIO de la carte Raspberry PI à l'aide des commandes shell.

2.2 Évaluation

L'évaluation portera sur :

- Un compte rendu détaillé explicitant les différentes opérations menées durant le TP. Pensez à prendre des copies d'écran pour illustrer ces opérations.
- Création d'une bibliothèque qui permet la manipulation des E/S GPIO de la carte Raspberry PI à l'aide des commandes Shell.
- Réalisation des deux manipulations (allumage d'une LED et un feu tricolore), pour se familiariser à utiliser notre propre bibliothèque pour la manipulation des E/S GPIO de la carte Raspberry PI dans le programme principal d'un projet.

2.3 Préparation de l'environnement de travail

2.3.1 Création d'un nouveau compte utilisateur

Nous allons utiliser la commande **adduser** pour créer les nouveaux comptes utilisateurs. La commande **adduser** est un script perl qui exécute une commande shell.

Quand elle est invoquée sans l'option **-D**, la commande *adduser* crée un nouveau compte utilisateur qui utilise les valeurs indiquées sur la ligne de commande et les valeurs par défaut du système. En fonction des options de la ligne de commande, la commande *adduser* fera la mise à jour des fichiers du système, elle pourra créer le répertoire personnel et copier les fichiers initiaux. Source : page **man** de **adduser**.

Syntaxe de la commande **adduser** :

adduser [*options*] **identifiant**

La figure 2.1 illustre l'ensemble des commandes pour ajouter un nouveau utilisateur selon les étapes suivantes :

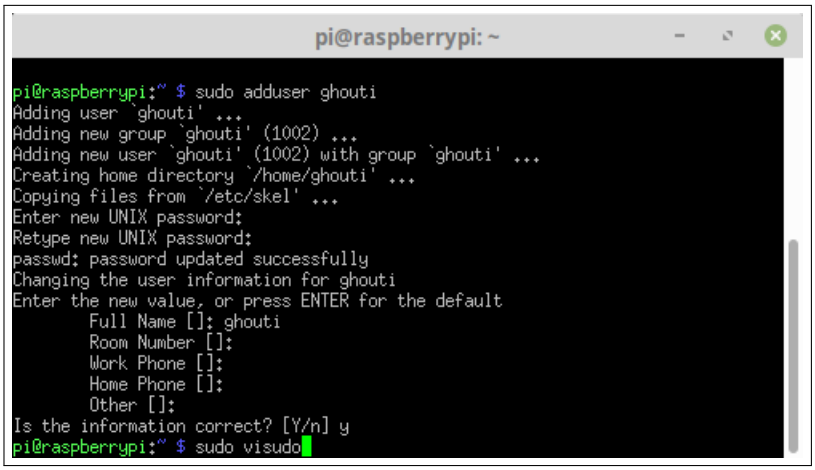


FIGURE 2.1: Ligne de commande pour ajouter un nouveau utilisateur

- Utiliser le logiciel PuTTY pour ouvrir une session SSH avec utilisateur **pi** et le mot de passe **raspberry**.
- Créer un nouvel utilisateur avec **votre nom** à l'aide de la commande **adduser** comme le montre la figure 2.1.
- Entrer le mot de passe pour cet nouvel utilisateur.
- Confirmer le nouveau mot de passe en le ressaisissant une deuxième fois.
- Entrer vos informations personnelles pour compléter la création du nouveau compte utilisateur comme le montre la figure 2.1.
- Utiliser la commande **sudo visudo** pour ajouter le nouveau utilisateur au groupe *Sudoers* (administrateur) en ajoutant la ligne suivante :

```
# User privilege specification
root    ALL=(ALL:ALL) ALL
ghouti  ALL=NOPASSWD:ALL
```

- Quitter l'éditeur de text à l'aide des touches clavier **ctrl+x**.
- Utiliser la commande **exit** pour fermer la session **pi**.
- Utiliser **PuTTY** pour ouvrir une nouvelle session SSH avec le nouveau utilisateur comme le montre la figure 2.2.

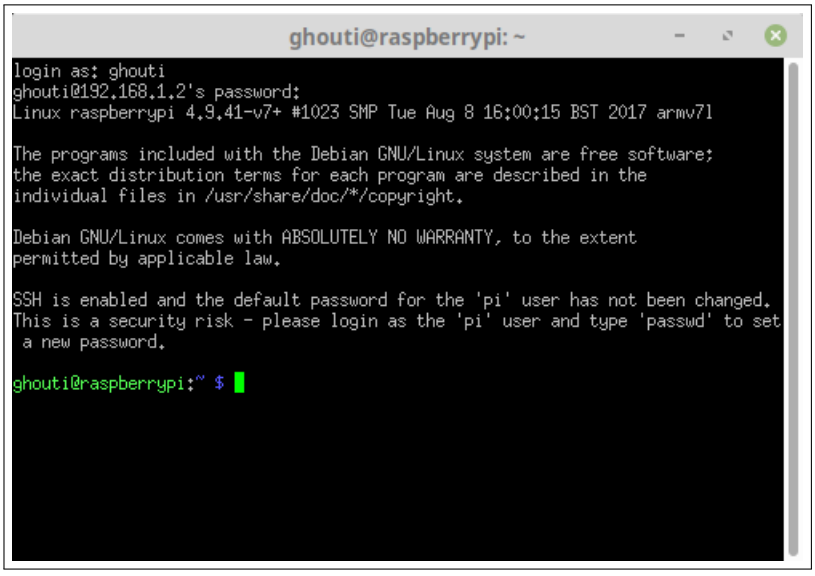


FIGURE 2.2: Connexion SSH avec le nouveau utilisateur

2.3.2 Préparation de la bibliothèque

Afin de développer notre propre bibliothèque C, nous allons utiliser l'éditeur de text **nano** pour créer un fichier nommé **gpio.h**, dans le quel nous allons écrire tous les fonctions nécessaires pour la manipulation des broches GPIO de la carte Raspberry Pi.

2.3.3 Travail demandé

- Écrire une fonction nommée *ConfigureGPIO* qui permet de Configurer une broche du port GPIO notée par la variable **Port** comme entrée ou sortie.

  void **ConfigureGPIO**(int Port,char IO[4]);

- Écrire une fonction nommée *UnConfigureGPIO* qui permet de supprimer la configuration.

  void **UnConfigureGPIO**(int Port);

- Écrire une fonction nommée *WritePort* qui permet d'écrire une valeur (1ou 0).

  void **WritePort**(int Port,int Val);

- Écrire une fonction nommée *ReadPort* qui permet d'écrire une valeur (1ou 0).

  char **ReadPort**(int Port);

Remarques *Utiliser une variable **cmd** de type chaîne de caractères pour stocker les commandes.*

*Utiliser la fonction **system** pour exécuter la commande stockée dans la variable cmd :*

*Exemple : **system**(cmd);*

Pour la manipulation de la variable cmd nous allons utiliser la fonction sprintf(), comme le montre la figure 2.3

```c
#include <stdio.h>
#include <math.h>
int main () {
   char str[80];
   sprintf(str, "Value of Pi = %f", M_PI);
   puts(str);
   return(0);
}
```

```
Apres l'exécution nous avons :

Value of Pi = 3.141593
```

FIGURE 2.3: Exemple d'utilisation de l'instruction sprintf()

2.4 Réalisation et test

Dans ce qui suit, nous allons réaliser deux applications basiques à base du RaspberryPI B version 3, ces exemples vont nous permettre de bien se familiariser avec l'utilisation de notre propre bibliothèque pour la manipulation des E/S GPIO de la RaspberryPI dans le programme principal.

2.4.1 Application N°1 : Clignotement d'une LED

Dans cette application, nous allons écrire un programme qui nous permet de clignoter la LED. Le schéma électronique ainsi que le cahier de charge est décrit comme suite.

Base d'expérimentation

Pour notre expérimentation, il nous faut :

- Une carte Raspberry Pi opérationnelle.
- Une LED.
- Une résistance de 270 Ω.
- Plaque d'essai.
- Des câbles.

Schéma électronique

La figure 2.4 montre le schéma électronique de la première application, L'anode de la diode LED (patte longue) est raccordée à la sortie GPIO17 (broche 11). La cathode (patte courte) est

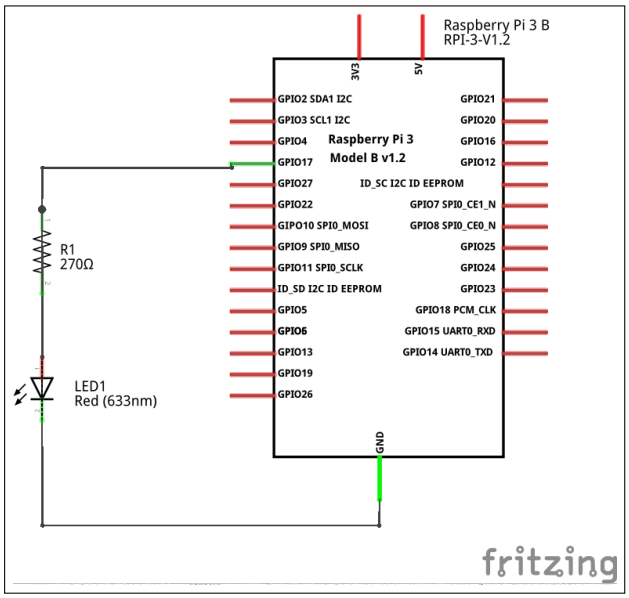


FIGURE 2.4: Schéma électronique avec une seule LED

reliée à la masse de la carte Raspberry Pi (Ground), par exemple la broche 6.

Schéma de cablage

Le montage de la première application est illustré par la figure 2.5 qui explique la façon de connecter les composants avec la carte Raspberry Pi selon le schéma de la figure 2.4.

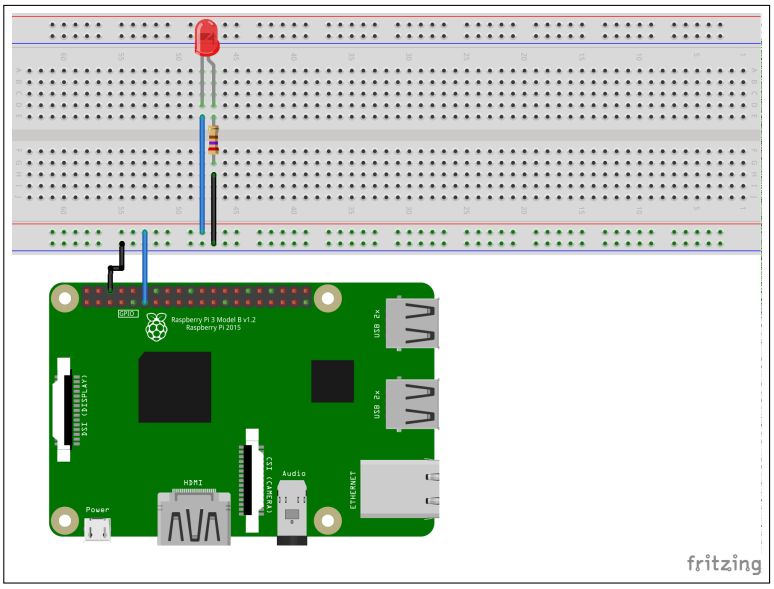


FIGURE 2.5: Schéma de câblage avec une seule LED

Cahier des charges

Tout programmeur qui se respecte doit disposer ou établir un cahier des charges qui définie les différentes opérations à réaliser. Dans notre cas, il sera très simple notre cahier des charges :

1. Allumer la LED,
2. Attendre 3 secondes,
3. Éteindre la LED,
4. Attendre 3 secondes,
5. Retour a 1 (on répète indéfiniment la boucle)

Travail demandé

Réaliser le montage illustré par la figure 2.5, puis utiliser l'éditeur **nano** pour écrire un programme qui répond au cahier des charges en faisant appel à la bibliothèque **gpio.h**

2.4.2 Application N°2 : Commande d'un feu tricolore

Le but de cette application est de réaliser la commande d'un seul feu tricolore. Les bases d'expérimentation, schéma électronique et le cahier des charges sont décrits comme suite :

Base d'expérimentation

Pour notre expérimentation, il nous faut :

- Une carte Raspberry Pi opérationnelle,
- Trois LEDs,
- Trois résistances de 270 Ω.
- Plaque d'essai.
- Des câbles.

Schéma électronique

Pour ce montage, nous allons utiliser trois LED de différentes couleurs (rouge, jaune et verte) pour faire un prototype d'un feu tricolore.

La figure 2.6 illustre le schéma électronique adéquat.

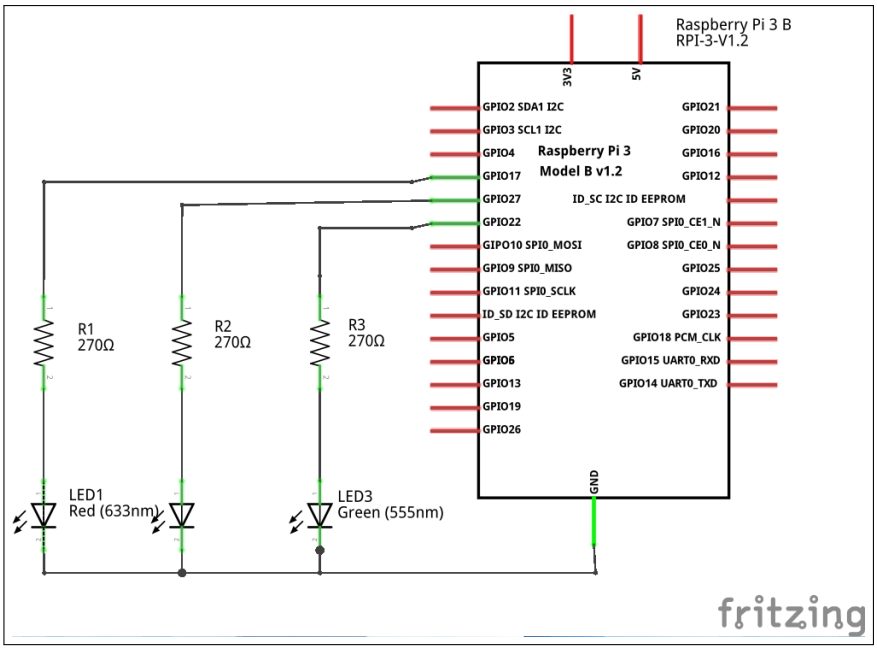


FIGURE 2.6: Schéma électronique pour un prototype d'un feux tricolore

Schéma de câblage

La figure 2.7 illustre la disposition et la manière de connecter les différents composants électronique sur la plaque d'essai conformément au schéma électronique illustré par la figure 2.6.

Les anodes des diodes LEDs (patte longue) sont raccordées aux sorties : GPIO17 (broche 11), GPIO27 (broche 13) et GPIO22 (broche 15).

Les cathodes (patte courte) sont reliées à la masse de la carte Raspberry Pi (Ground), par exemple la broche 6.

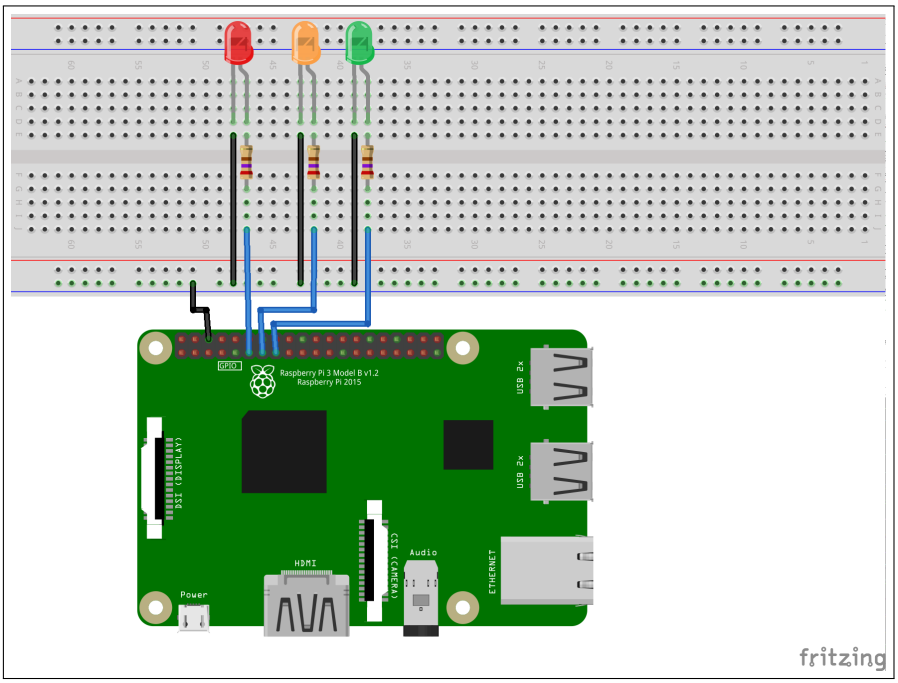


FIGURE 2.7: Schéma de câblage pour un prototype d'un feux tricolore

Cahier des charges

Le fonctionnement du feu tricolore est détaillé comme suit :

1. Allumer le feu vert pendant 30 secondes.
2. Allumer le feu orange pendant 05 secondes.
3. Allumer le feu rouge pendant 20 secondes.
4. Reprendre le cycle du début.

Travail demandé

Réaliser le montage illustré par la figure 2.7, puis utiliser l'éditeur **nano** pour écrire un programme qui répond au cahier des charges en faisant appel à la bibliothèque **gpio.h**

2.5 Conclusion

Linux est un système d'exploitation qui se base principalement sur les fichiers, dans ce TP nous avons manipulé les fichiers système afin de commander les ports GPIO en utilisant des fonctions écrites en langage C. Dans la littérature, il existe plusieurs bibliothèques prédéfinies et open source pour manipuler les port GPIO tel que *WringPi*.

TP N° 3

Commande d'un moteur à courant continu avec Raspberry Pi en C

3.1 Introduction

Durant ce TP, nous allons contrôler la vitesse d'un moteur à courant continu à l'aide d'une carte Raspberry Pi et la technique PWM. La modulation de largeur d'impulsion (PWM) est une méthode utilisée pour obtenir une tension variable en dehors d'une source d'alimentation constante.

3.2 Évaluation

L'évaluation portera sur :

- Un compte rendu détaillé explicitant les différentes opérations menées durant le TP. Pensez à prendre des copies d'écran pour illustrer ces opérations.
- Réalisations du circuit électronique qui offre la possibilité à la Raspberry Pi de commander le moteur.
- Implémentation d'un programme en C qui permet de contrôler le moteur.

3.3 Principe du fonctionnement du moteur

Un moteur à courant continu en anglais *direct current* (DC) est une machine électrique. Il s'agit d'un convertisseur électromécanique permettant la conversion bidirectionnelle d'énergie entre une installation électrique parcourue par un courant continu et un dispositif mécanique.

3.4 Réalisation et test

Dans ce qui suit, nous allons réaliser un simple schéma électronique qui permet de contrôler un moteur DC a l'aide des entrées sorties GPIO

3.4.1 Application N°1 : Démarrage et arrêt d'un moteur Dc

Dans cette application, nous allons écrire un programme qui nous permet de faire une commande simple d'un moteur DC. Le schéma électronique ainsi que le cahier de charge est décrit comme suite.

Base d'expérimentation

Pour notre expérimentation, il nous faut :

- Une carte Raspberry Pi opérationnelle.
- Un petit moteur à courant continu.
- Une résistance de 220 Ω.
- Un transistor 2N2222.
- Une diode 1N4007.
- Un condensateur 1000 μF.
- Une plaque d'essai.
- Des broches de connexion.

Schéma électronique

La figure 3.1 illustre notre schéma électronique pour l'application N°1, sachant que nous ne pouvons pas tirer plus de **50 mA** de toutes les broches **GPIO** et le moteur à courant continu exige plus de **50 mA**. Donc, si nous connectons le moteur directement à la carte Raspberry Pi pour contrôler la vitesse, cette dernière risque d'être endommagée de manière permanente.

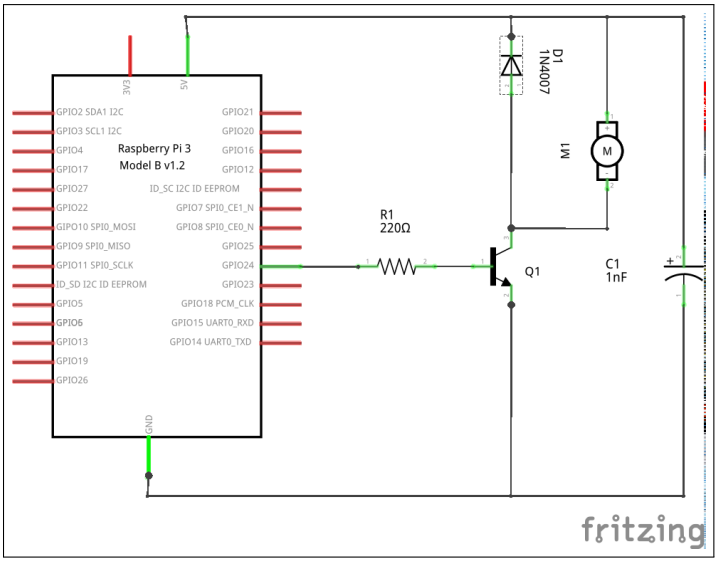


FIGURE 3.1: Schéma électronique pour un moteur Dc

Nous allons donc utiliser un transistor **NPN** (**2N2222**) comme dispositif de commutation . Ici, ce transistor pilote le moteur à courant continu haute puissance en prenant le signal PWM de la carte Raspberry Pi. Ici, il faut faire attention au fait que la connexion incorrecte du transistor peut charger fortement la carte.

Le moteur est une induction et lors de la commutation du moteur, nous faisons l'expérience de pics inductifs. Ce picage chauffera fortement le transistor, nous allons donc utiliser la **diode** (**1N4007**) pour protéger le transistor contre le picage inductif.

Afin de réduire **les fluctuations de tension** , nous allons connecter **un condensateur** de **1000** μF à l'alimentation électrique, comme indiqué dans le schéma de circuit.

Schéma de câblage

Dans la figure 3.2, nous allons montrer la façon de faire le câblage adéquat au schéma électronique illustré par la figure 3.1.

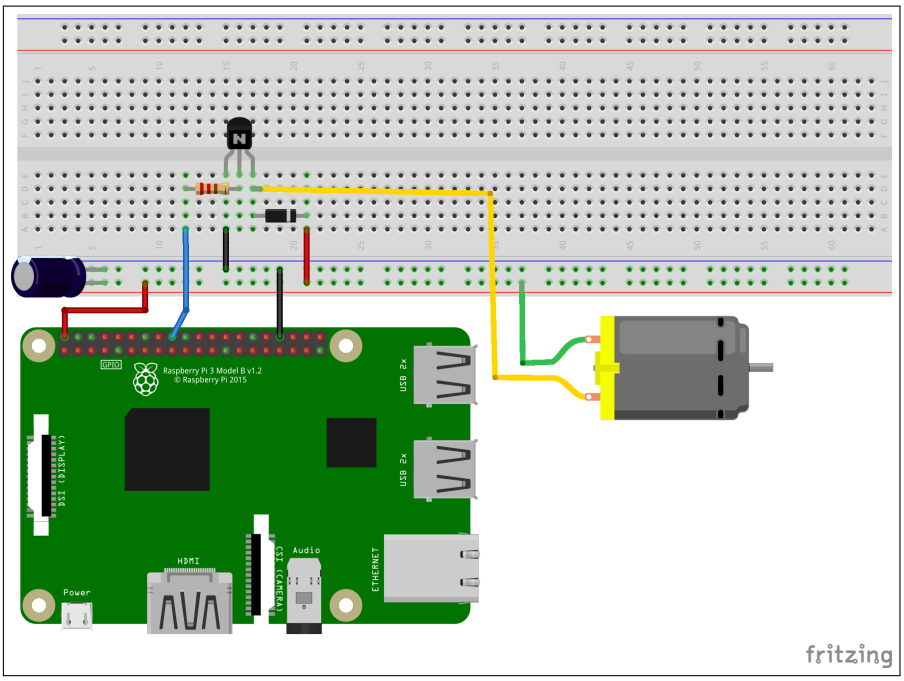


FIGURE 3.2: Schéma de câblage pour un moteur Dc

Cahier des charges

Une fois que tout est connecté conformément au schéma de câblage illustré par la figure 3.2, nous pouvons se connecter à la carte Raspberry PI par le biais d'un micro-ordinateur, afin d'écrire un programme C qui doit répondre au cahier des charges suivant :

1. Configurer le GPIO 24 en sortie,
2. Faire tourné le moteur,
3. Attendre 10 secondes,
4. Arrêter le moteur,
5. Désactiver le port GPIO 24.

3.4.2 Application N°2 : Variation de la vitesse d'un moteur Dc

Dans cette application, nous allons écrire un programme qui nous permet de contrôler la vitesse du moteur en utilisant la technique PWM (Pulse Wide Modulation) ou La modulation de la largeur d'impulsions. Le schéma électronique ainsi que le cahier de charge est décrit comme suite.

Base d'expérimentation

Pour cette deuxième application nous allons utiliser la même base d'expérimentation.

Schéma électronique

Voir schéma électronique de la première application illustré par la figure 3.1.

Principe de fonctionnement du PWM

La PWM est une technique couramment utilisée pour synthétiser des signaux pseudo analogiques à l'aide de circuits à fonctionnement tout ou rien, ou plus généralement à états discrets.

La technique PWM se repose sur deux paramètres de base :

- **la période du cycle (period)** : Le temps de cycle du début d'une impulsion à une autre.
- **le rapport cyclique(duty cycle)** : C'est un pourcentage du temps que le prend l'impulsion par rapport à la période.

La figure 3.3 illustre quelques chronogramme pour des différentes valeurs du rapport cyclique.

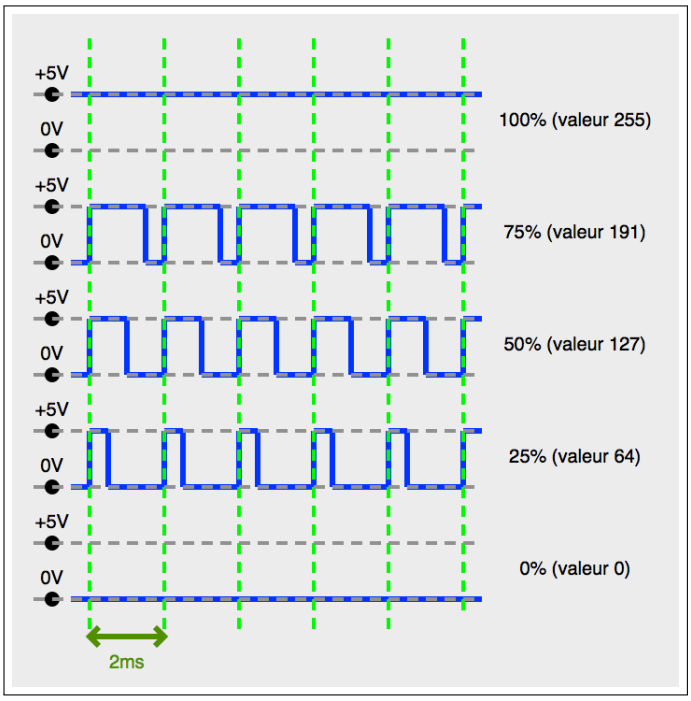


FIGURE 3.3: Chronogramme pour des différentes valeurs du rapport cyclique

Principe de fonctionnement de la PWM sous la Raspberry Pi

La PWM sur la carte Raspberry Pi est une deuxième fonction pour certains ports GPIO, elle est désactiver par défaut pour l'activer il faut suivre les étapes suivantes :

1. Utiliser l'éditeur **nano** en mode super utilisateur (sudo), pour ouvrir et éditer le fichier *config.txt* à l'aide de la commande suivante :

   $sudo /boot/config.txt

2. Ajouter la linge de configuration suivante : **dtoverlay=pwm-2chan**.
3. Enregistre les modifications sur le fichier *config.txt* à l'aide des touches clavier **ctrl+o**.
4. Quitter l'éditeur à l'aide des touches clavier **ctrl+x**.
5. Redémarrer la carte Raspberry Pi.

Après avoir redémarré votre carte Raspberry Pi, vous aurez accès au matériel PWM sur les broches 12, 32 et 33.

Vous trouverez un nouveau répertoire nommé *pwmchip0* (**/sys/class/pwm/pwmchip0**), qui fonctionne de la même manière que le support du système de fichiers pour GPIO : il existe une exportation d'un fichier bien déterminé, que vous utilisez pour accéder aux broches PWN :

- echo 0 > export : broches 12 et 32.
- echo 1 > export : broche 33.

L'exécution de ces deux commandes entraînera l'apparition de deux nouveaux répertoires :

/sys/class/pwm/pwmchip0/pwm0

/sys/class/pwm/pwmchip0/pwm1.

Chacun de ces répertoires contient des fichiers spéciaux permettant de contrôler la sortie PWM :

- **duty_cycle** : définit le rapport cyclique du signal PWM.
- **enable** : active (écrit un 1) ou désactive (écrit un 0) une sortie PWM.
- **period** : définit la période du signal PWM.

Duty_cycle et **period** s'attendent tous les deux à des valeurs **entières** et en **nanosecondes**. Ainsi, par exemple, pour émettre une tonalité à 440Hz, vous devez d'abord calculer la période correspondant à cette fréquence :

$$period = \frac{1}{freq} = \frac{1}{440} = 0,002272727 * 10^9 s = 2272720 ns$$

Pour avoir 50% duty cycle, il suffit juste de diviser la période par 2 :

$$duty\_cycle = periode/2 = 2272720/2 = 1136360$$

Maintenant, faites écho ces valeurs aux fichiers appropriés :

echo "2272720" > /sys/class/pwm/pwmchip0/pwm0/period

echo "1136360" > /sys/class/pwm/pwmchip0/pwm0/duty_cycle

En résumé, si nous voulons utiliser la broche 12 pour activer la pwm avec une période 440Hz et 50% duty cycle nous devons passer par les étapes suivantes :

- Configurer la broche 12 en pwm : echo "0">/sys/class/pwm/pwmchip0/export
- Configurer la période : echo "2272720" > /sys/class/pwm/pwmchip0/pwm0/period
- Configurer le rapport cyclique : echo "1136360" > /sys/class/pwm/pwmchip0/pwm0/duty_cycle
- Activer la broche 12 : echo "1" > /sys/class/pwm/pwmchip0/pwm0/enable

Cahier des charges

Une fois que tout est connecté conformément au schéma de circuit, nous pouvons utiliser la RaspberryPi et écrire le programme en C selon le cahier des charges suivant :

- Démarrer le moteur,
- Attendre 10 secondes,
- Arrêter le moteur,

Afin de répondre a ce cahier des charges, nous devons suivre la démarche suivante :

1. Dans la bibliothèque (**gpio.h**) ajouter les fonctions de configuration et manipulation des pins PWM, et qui ont les prototypes suivants :
   - Une fonction qui permet d'exporter un port PWM (0 ou 1) en lui attribuant une fréquence Freq :

     ConfigurePWM(int Port,float Freq);
   - Une fonction qui permet de Configurer un port PWM avec un certain rapport cyclique DtC :

     SetDutyCycle(int Port, float Freq,float DtC);
   - Une fonction qui permet d'activer un port PWM

     SetEnablePWM(int Port,int Val);
   - Une dernière fonction qui permet d'annuler l'utilisation d'un port PWM :

     UnConfigurePWM(int Port);
2. Écrire un programme (**tp3.c**) qui permet de tourner le moteur DC,

Valeurs des arguments utilisés pour les fonctions :

- **int Port** : prend la valeur 0 ou 1 pour exporter respectivement la PWM0 ou PWM1.

- **float Freq** : prend la valeur de la fréquence en Hz.
- **float Dtc** : représente le pourcentage du rapport cyclique.
- **int Val** : prend la valeur 0 ou 1 pour activer ou désactiver la PWM

3.5 Conclusion

Nous savons maintenant comment manipuler les fichiers système afin d'utiliser la technique PWM pour faire varier la puissance d'un signal en utilisant des fonctions écrites C. Dans la littérature, il existe plusieurs bibliothèques préprogrammées et open source pour manipuler les port GPIO tel que *WringPi*.

TP N° 4

Régulation de la température ambiante à base d'un moteur DC

4.1 Introduction

Plusieurs systèmes industriels exigent un contrôle automatisé de la température tels que (température d'une cuve, d'un four, ...) ou domestique (climatisation d'une salle, d'une voiture, ...).

La réalisation d'un tel système, exige un contrôle en temps réel de la vitesse d'un ventilateur en fonction de la température, ceci peut être vue comme une fonction de régulation d'un système aérotherme constitué d'un ventilateur (Moteur), une source de chaleur et un capteur de température. L'entrée du système est la tension au borne du moteur et la sortie sera la température mesurée par le capteur.

4.2 Évaluation

L 'évaluation portera sur :

— Un compte rendu détaillé explicitant les différentes opérations menées durant le TP. Pensez à prendre des copies d'écran pour illustrer ces opérations.

— Réalisations du circuit électronique qui offre la possibilité a la Raspberry Pi de mesurer la température et commander le moteur.

— Implémentation d'un programme en C qui permet de contrôler le moteur.

4.3 La régulation Proportionnelle (P)

En régulation, la valeur de la mesure est désignée par la lettre X et la valeur de la consigne par la lettre W. Sur notre schéma général (voir la figure 4.1), nous allons utiliser la lettre C pour la consigne et la lettre M pour la mesure. Sur un régulateur, la consigne sera désignée par les lettres SP et la mesure par les lettres PV. La figure 4.1 illustre un schéma général d'un système de régulation qui sera décomposé en deux bloc principaux :

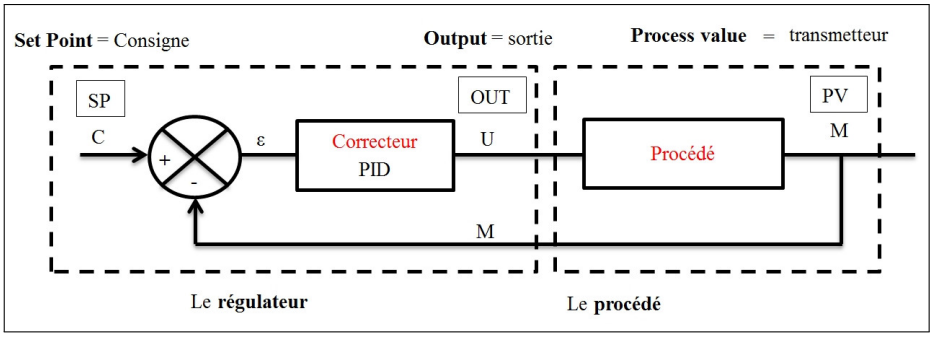


FIGURE 4.1: Schéma général d'un système de régulation

— **Le procédé** est le système physique sur lequel est placée (en sortie) la grandeur physique à asservir notée M(mesure). Sa grandeur d'entrée U est aussi la sortie du correcteur. C'est une grandeur électrique (tension ou courant).

— **Le correcteur** : À partir de l'écart ε entre la consigne et la mesure, il génère le signal de commande U qui agit sur la variable réglante du procédé.

  En régulation le correcteur de base est le PID. Ce correcteur est composé de 3 fonctions :

  — la fonction proportionnelle

  — la fonction intégrale

  — la fonction dérivée

Dans notre cas nous allons étudier un système de **régulation proportionnelle** qui peut être définit comme suite :

Définition *L'action Proportionnelle corrige de manière instantanée, donc rapide, tout écart de la grandeur à régler, elle permet de vaincre les grandes inerties du système. Afin de diminuer l'écart de réglage et rendre le système plus rapide, on augmente le gain (on diminue la bande proportionnelle) mais, on est limité par la stabilité du système. Le régulateur P est utilisé lorsque on désire régler un paramètre dont la précision n'est pas importante, exemple : régler la température ambiante d'une salle.*

4.4 Réalisation et test

Pour réaliser notre système de régulation, nous allons décomposer notre démarche en deux étapes représentées par les deux applications suivantes :

- Application N°1 : Le contrôle de la vitesse du moteur DC à l'aide des boutons poussoirs.
- Application N°2 : L'utilisation du capteur de température pour effectuer la régulation de la température.

4.4.1 Application N°1 : Contrôle de la vitesse d'un moteur Dc à l'aide de deux boutons

Dans une première application, nous allons écrire un programme qui nous permet de contrôler un moteur DC à l'aide de deux boutons poussoirs, dont chacun parmi eux permet de faire tourner le moteur a une vitesse donnée. Le schéma électronique ainsi que le cahier des charges est décrit comme suite.

Base d'expérimentation

Pour notre expérimentation, il nous faut :

- Une carte Raspberry Pi opérationnelle.
- Un moteur à courant continu.

- Une résistance de 220 Ω.
- Deux résistances de 10 KΩ.
- Un transistor 2N2222.
- Une diode 1N4007.
- Un condensateur 1000μF.
- Une plaque d'essai.
- Deux boutons poussoirs.
- Des broches de connexion.

Schéma électronique

La figure 4.2, illustre le schéma électronique qui nous permet de contrôler le moteur Dc à l'aide des deux boutons poussoirs.

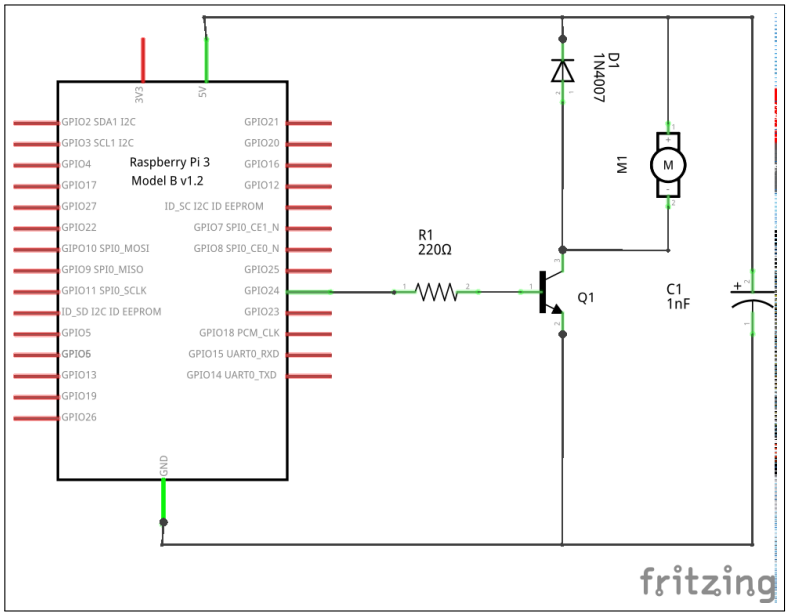


FIGURE 4.2: Schéma électronique pour commander un moteur Dc à l'aide de deux boutons

Schéma de câblage

La figure 4.3, illustre la disposition et la façon de raccorder les composants électronique sur un plaque d'essai conformément au schéma électronique illustré par la figure 4.2.

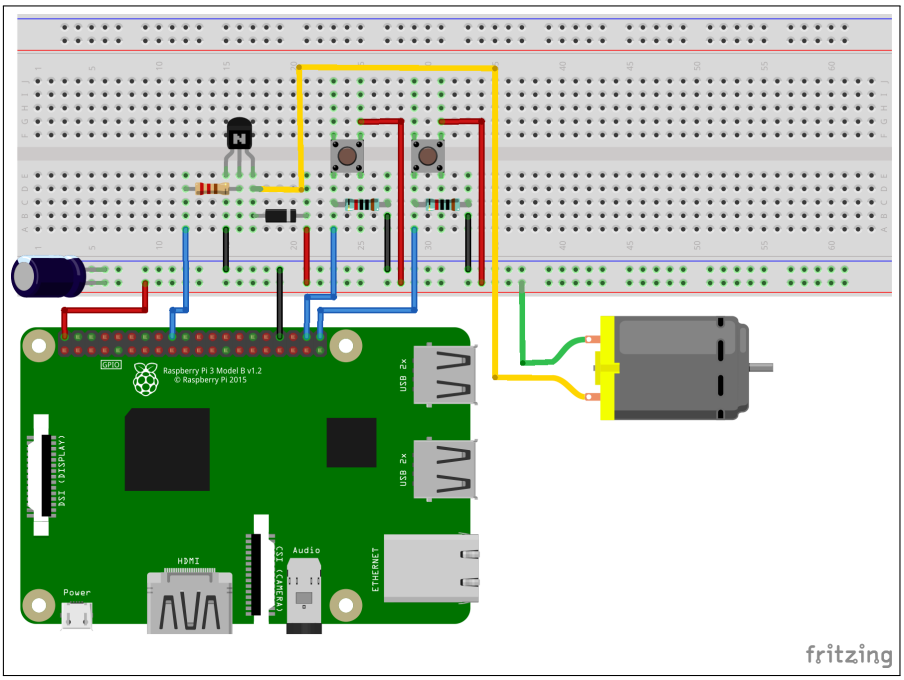


FIGURE 4.3: Schéma de câblage pour commander un moteur Dc à l'aide de deux boutons

Cahier des charges

Une fois que tout est connecté conformément au schéma de circuit, nous pouvons utiliser le PI pour écrire le programme C nommé (**tp4.c**) selon le cahier des charges suivant :

1. Créer deux fonctions (Vitesse1, Vitesse2) qui permettent d'attribuées les rapports cycliques 50% et 75% respectivement en utilisant la fonction **SetDutyCycle(int Port, float Freq,float DtC);**
2. Dans le programme principal, réaliser les étapes suivantes :
   - Configurer le PWM0 en utilisant la fonction **ConfigurePWM(int Port,float Freq)**
   - Attribuer un rapport cyclique de 25% en utilisant la fonction **SetDutyCycle(int Port, float Freq,float DtC)**
   - Démarrer le moteur en utilisant la fonction **SetEnablePWM(int Port,int Val)**,
   - Tant que vrai, attendre un événement du bouton1 ou bouton2.
   - Si les deux boutons serons actionnés au même temps Arrêter le moteur,
   - Désactiver la PWM en utilisant la fonction **UnConfigurePWM(int Port)**.

Remarque : Créer une nouvelle fonction dans le fichier **gpio.h** qui permet de lire une valeur d'un port GPIO comme le montre la figure 4.4.

```
#include<stdio.h>
#include <fcntl.h>
char ReadPort(char Port[3]){
char c;
char Path[80];
int file;
        sprintf(Path,"/sys/class/gpio/gpio%s/value",Port);
        file =  open(Path, O_RDONLY);
        if(file!=-1) read(file,&c,1);
        close(file);
return c;
}
```

FIGURE 4.4: Implémentation de la fonction ReadPort

4.4.2 Application N°2 : Régulation de la température à l'aide d'un moteur DC

Dans cette application, nous allons automatiser notre système de contrôle de température en remplaçant les deux boutons par un capteur de température.

Base d'expérimentation

Pour cette deuxième application nous allons utiliser les composants suivants :

- Une carte Raspberry Pi opérationnelle.
- Un moteur à courant continu.
- Une résistance de 220 Ω et deux résistances de 10 $k\Omega$.
- Un transistor 2N2222.
- Une diode 1N4007.
- Un condensateur 1000μF.
- Une plaque d'essai.
- Un bouton poussoir.
- Capteur de température.
- Des broches de connexion.

Schéma électronique

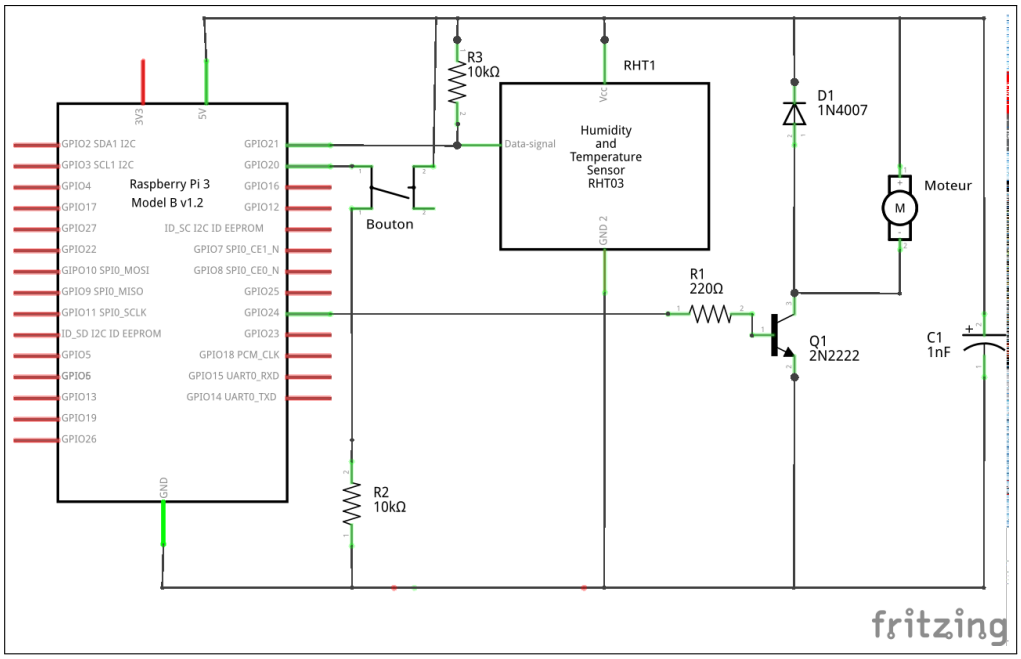


FIGURE 4.5: Schéma électronique de commande d'un moteur Dc à l'aide d'un capteur de température

Schéma de cablage

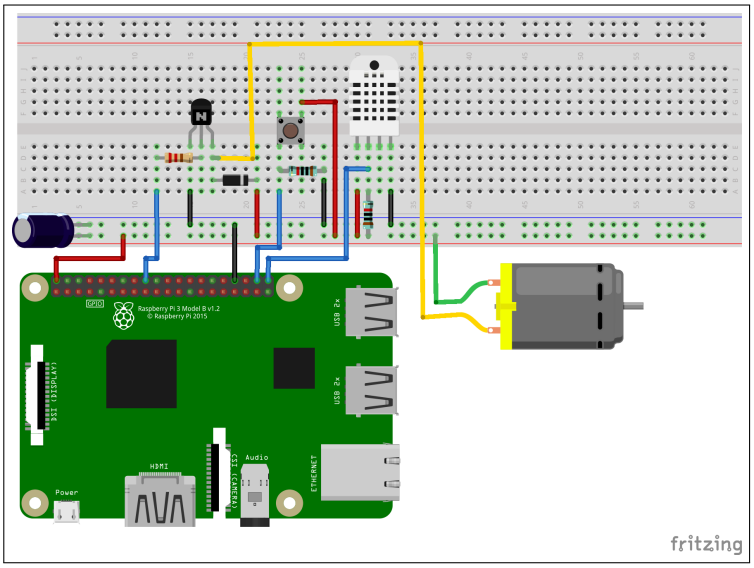


FIGURE 4.6: Schéma de câblage pour un moteur Dc commandé par un capteur de température

Cahier des charges

Une fois que tout est connecté conformément au schéma de circuit, nous pouvons utiliser la carte Raspberry PI pour écrire le programme en C selon le cahier des charges suivant :

1. Créer une fonctions nommée **ModifierVitesse(int Val)** qui permettent d'augmentée le rapport cyclique par un pas égale a **Val**.
2. Dans le programme principale réaliser les étapes suivantes :
   - Configurer le PWM0 en utilisant la fonction **ConfigurePWM(int Port,foat Freq)**
   - Attribuer un rapport cyclique de 10% par en utilisant fonction **SetDutyCycle(float DtC);**
   - Démarrer le moteur en utilisant la fonction **SetEnablePWM(int Val)**,
   - Tant que vrai, et selon l'écart de température entre la température mesurée et la consigne si :
     - l'écart est supérieur à 2° : augmenter la vitesse du moteur en utilisant la fonction **ModifierVitesse(int Val)**, avec Val=10.
     - l'écart est inférieur à -2° : diminuer la vitesse du moteur en utilisant la fonction **ModifierVitesse(int Val)**, avec Val=-10.

     sachant que le rapport cyclique doit être compris entre 10% et 90%
   - Si le bouton 1 sera actionné arrêter le moteur,
   - Désactiver la PWM en utilisant la fonction **UnConfigurePWM(int Port)**.

4.5 Conclusion

Durons ce TP nous avons réaliser un contrôle de température simplifier qui permet juste de faire le refroidissement d'une salle, pour compléter ce genre système il nous faut ajouter une source de chaleur par exemple une chaudière, qui sera commandée par la deuxième broche PWM (PWM1) en controlant l'ouverture et la fermeture de l'électrovanne afin de varier la quantité de gaz écoulé dans la chaudière et par conséquent augmenter la température ambiante.

TP N° 5

Transmission de données via des interfaces hertziennes

5.1 Introduction

La communication est une caractéristique indispensable dans les systèmes embarqués modernes, certains systèmes qui ne communiquent pas sont jugés inutiles car ils n'évoluent pas rapidement dans le temps. Heureusement la carte Raspberry Pi est dotée de plusieurs supports de communication tels que les ports GPIO(I2C), WIFI, Bluetooth,etc. Dans ce TP nous allons intégrer un module de communication RF (Radio Fréquence), qui utilise la fréquence **433Mhz** pour transmettre et recevoir les données.

5.2 Évaluation

L 'évaluation portera sur :

- Un compte rendu détaillé explicitant les différentes opérations menées durant le TP. Pensez à prendre des copies d'écran pour illustrer ces opérations.
- Réalisations du circuit électronique qui offre la possibilité à la Raspberry Pi de communiquer a travers une interface hertzienne (Antenne).

— Implémentation d'un programme en C qui permet de transmettre des données vers une autre carte Raspberry Pi.

5.3 Communication Hertzienne

Les communications hertzienne ou radiocom est un résultat de l'exploitation du spectre électromagnétique en champ libre pour effectuer de la transmission d'information. Ceci sera utile pour plusieurs applications qui se basent sur les principes suivants :

— La diffusion.

— La liaison point à point.

— L'accès mobile.

— Les radio-mobile.

— Les réseaux cellulaires.

— Le contrôle à distance.

— ...

5.3.1 La bande de fréquence 433Mhz

Le 433Mhz est un ensemble de protocole de communications radio utilisé généralement dans les applications à faible porté tels que le contrôle domotique :

— Une sonnette sans fil et son bouton poussoir.

— Des prises commandés et leur télécommande.

— Une station météo et ses sondes de température.

— Portail et porte de garage automatisée.

— les commande d'ouverture et fermeture centralisé dans l'automobile.

— etc.

L'avantage de cette fréquence est qu'elle est libre d'utilisation, et déjà largement utilisée pour des communications à courte portée.

5.3.2 Les modules radio 433MHz

Ce sont des modules de transmission radio extrêmement simpliste comme le montre la figure 5.1, ils utilisent la modulation ASK (Amplitude Shift Keying)(l'équivalent numérique de la modulation d'amplitude analogique (radio **AM**)) et qui contiennent le strict nécessaire pour effectuer une transmission, ce qui explique leurs faible prix. Cependant, bien extrêmement simplistes, ces modules peuvent être la solution idéale en fonction du projet.

Ces modules radio sont disponibles en deux variantes :

— Émetteur figure 5.1 (a) : pour envoyer des messages.

— Récepteur figure 5.1 (b) : pour recevoir des messages

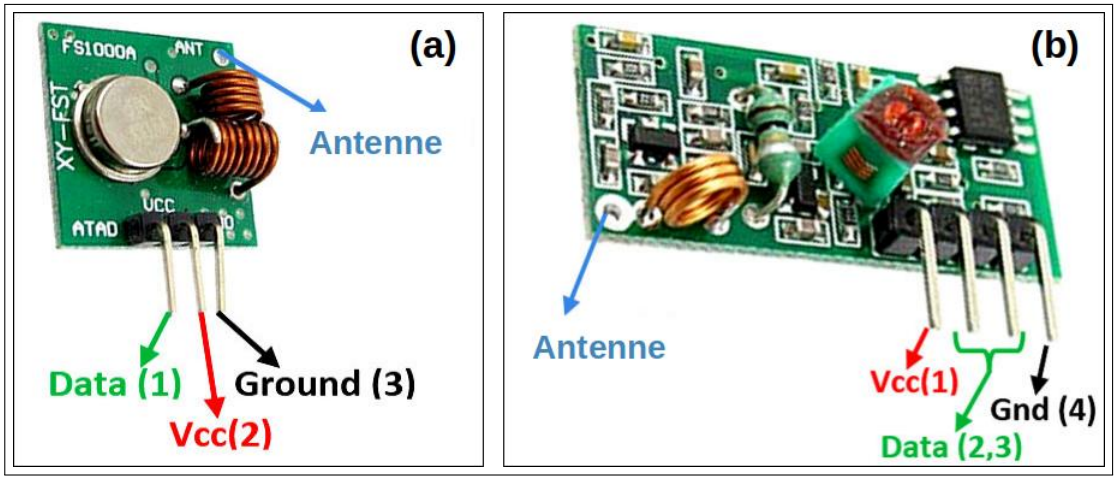


FIGURE 5.1: Modules radio 433MHz

Ces modules ne peuvent être utilisés que par paires et seule une communication simplex est possible. Cela signifie que l'émetteur ne peut transmettre que des informations et que le récepteur ne peut que les recevoir.

Branchement sur la carte Raspberry Pi

Pour le récepteur :

— GND : Sur le GND de la carte Raspberry Pi.

— VCC : Sur le 5v de la carte Raspberry Pi.

— Data : Sur le port GPIO 27 de la carte Raspberry Pi.

Pour l'émetteur (le plus petit avec 3 bornes) :

— GND : Sur le GND de la carte Raspberry Pi.

— VCC : Sur le 5v de la carte Raspberry Pi.

— Data : Sur le port GPIO 17 de la carte Raspberry Pi.

5.4 Réalisations et tests

Afin de comprendre le principe de la communication nous allons réaliser deux montages avec la carte Raspberry Pi, dont le premier représente une communication avec la même carte Raspberry Pi et la deuxième entre deux cartes Raspberry Pi différentes.

5.4.1 Application N°1 : Communication entre l'émetteur et le récepteur de la même carte Raspberry Pi

Cette première application consiste à réaliser un simple montage avec une seule carte Raspberry Pi et deux antennes (émettrice, réceptrice), et écrire un programme en C qui permet de tester la communication entre ces deux antennes.

Base expérimentale

— Une carte Raspberry Pi opérationnelle.

— Une antenne émettrice Tx.

— Une antenne réceptrice Rx.

Schéma électronique

La figure 5.2, illustre le schéma électronique pour connecter les module RF avec la carte Raspberry Pi.

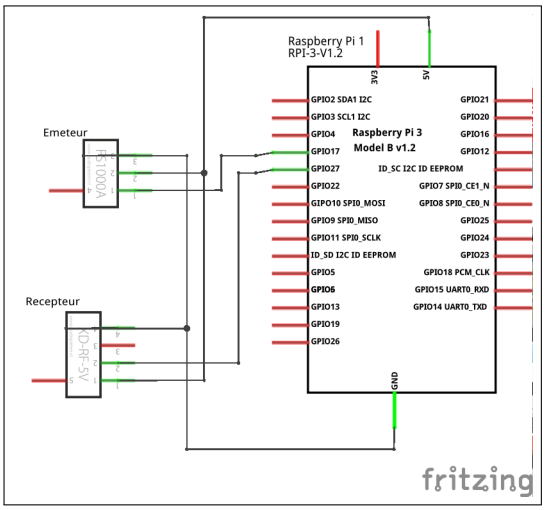


FIGURE 5.2: Schéma électronique d'une carte Raspberry Pi avec deux antennes Tx,Rx

Schéma de câblage

Sur votre plaque d'essai, reporter le même schéma de câblage illustré par la figure 5.3

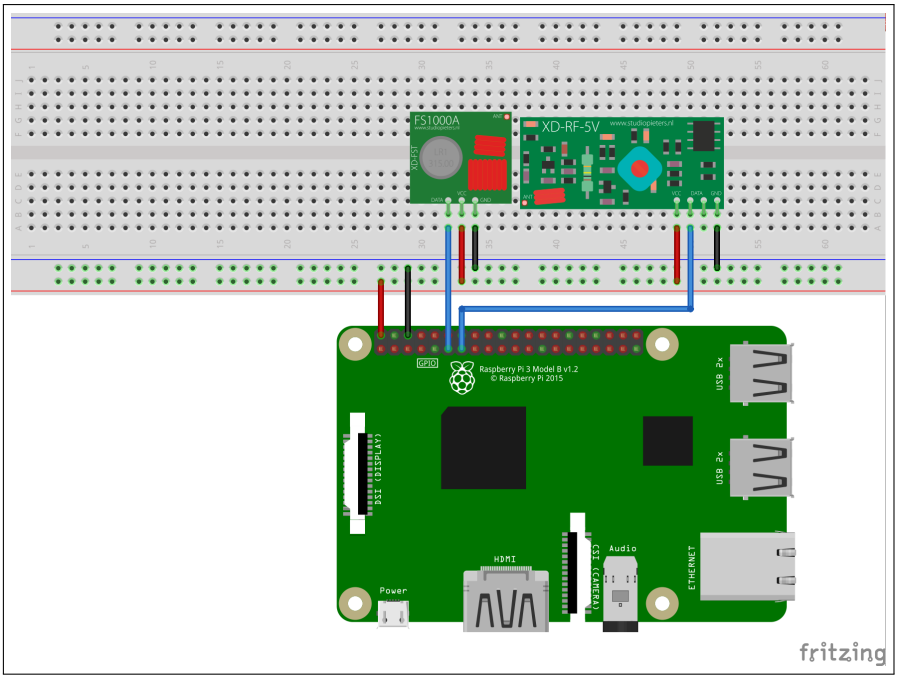


FIGURE 5.3: Schéma de câblage d'une carte Raspberry Pi avec deux antennes Tx,Rx

Cahier des charges

Une fois que tout est connecté conformément au schéma de circuit, nous pouvons utiliser la carte Raspberry Pi pour écrire un programme en C qui permet d'effectuer une communication interne (sur la même carte Raspberry Pi en passant par les étapes suivantes :

1. Écrire deux programmes en C qui permettent de :
   - Le premier programme : Envoyer une chaine de caractère en faisant appel au programme **codeSend.x**.
   - Le deuxième programme : Recevoir une chaine de caractère en faisant appel au programme **RFSniffer.x**.
2. Écrire un programme en C qui permet de créer deux Threads le premier pour l'envoie des données et le deuxième pour la réception des données.

Remarque : Utiliser l'une des commande de la famille **exec** pour éxecuter les programmes *CodeSend.x* et *RFSniffer.x*.

5.4.2 Application N°2 : Allumage distant d'une LED à l'aide d'une communication RF

Dans cette partie, nous allons utiliser deux cartes Raspberry Pi, la première effectue le contrôle de la LED à l'aide d'un bouton poussoir et un émeteur RF. La deuxième éxécute la commande d'allumage de la LED à l'aide d'une LED bien-sur et un récepteur RF.

Pour l'utilisation des deux modules RF, nous allons faire appel à la bibliothèque **433Utils** qui peut être téléchargée et installée a partir du dépôt :

https ://github.com/ninjablocks/433Utils.git

En faisant appel aussi à la bibliothèque **wiringpi**, téléchargeable et installable à partir du dépôt :

git ://git.drogon.net/wiringPi

Base expérimentale

Pour cette application nous allons avoir besoin des équipements suivants :

- Deux Cartes Raspberry Pi opérationnelles.
- Un module radio fréquence émeteur.
- Un module radio fréquence récepteur.
- Une LED.
- Une résistance 220Ω.
- Une plaque d'essai.
- Des câbles de raccordement.

Cahier des charges

Dans cette application nous allons vous demander de faire vous même le schéma électronique qui permet d'expliquer le brochage de tous les composants électronique cités dans la base expérimentale.

Dans une seconde étape, nous allons vous demander de réaliser le montage adéquat a votre schéma électronique.

Et comme dernière étape, nous allons vous demander d'écrire des programmes en C qui permettent d'effectuer cette commande à distance de la LED.

5.5 Conclusion

Difficile d'imaginer le monde actuel sans communication sans-fil. La plupart des technologies sans-fil fonctionnent avec la transmission et la modulation de fréquence, les données numériques étant converties en signaux radio afin d'être transmises vers d'autres périphériques de contrôle.

Le module sans fil à 433 MHz est l'un des modules où le coût est le plus moins cher dans le marché vue qu'ils sont les plus faciles à utiliser pour tous les projets sans fil.

Le module peut couvrir un minimum de 3 mètres et avec une antenne appropriée, une alimentation électrique peut atteindre théoriquement jusqu'à 100 mètres. Mais dans la pratique, nous pouvons difficilement atteindre environ 30 à 35 mètres dans des conditions de test normales.

Conclusion

L'apparition des systèmes embarqués dans le monde industriel a changé la manière de concevoir les systèmes technologique en introduisant la notion de la programmation aux circuits électroniques, ceci a simplifié la conception et la réalisation des systèmes en déplaçant la complexité hardware vers une solution software.

Pour cette raison, et durons tous les travaux pratiques nous avons étudier les principaux applications de base pour l'utilisation d'un système embarqué, commençons par l'utilisation des entrées sorties de la carte Raspberry Pi, passons par la commande numérique d'un équipement (moteur), nous avons aussi étudier les principes de la technique PWM par une application pratique, jusqu'au la communication radio fréquence. Tous ces travaux ont concrétisés nos connaissances sur les systèmes embarqués avec système d'exploitation.

Références

— BLAESS, Christophe. Programmation système en C sous Linux : Signaux, processus, threads, IPC et sockets. Editions Eyrolles, 2005.

— BLAESS, Christophe. Scripts shell Linux et Unix : Avec 30 scripts prêts à l'emploi. Editions Eyrolles, 2012.

— FICHEUX, Pierre et BÉNARD, Eric. Linux embarqué : Nouvelle étude de cas-Traite d'OpenEmbedded. Editions Eyrolles, 2012.

— KIRSTEN, Kearney et WILL Freeman.Raspberry Pi : 35 projets ludiques.Editions Dunod, 2018.

— METTIER, Yves. C en action. Editions ENI, 2009.

— SARTOR, Kevin. Raspberry Pi et ESP8266-Domotisez votre habitation. Editions ENI, 2018.

Annexes

Liste des abréviations

Abréviation	Définition
Carte SD	Carte Secure Digital
CPU	Central Processing Unit
DC	Direct Current
FS	File System
GPIO	General Purpose Input/Output
GUI	Graphical User Interface
LED	Light Emitting Diode
PWM	Pulse Wide Modulation
RF	Radio Frequency
SE	Système d'Exploitation
SSH	Secure Shell
TOR	Tous Ou Rien
TP	Travaux Pratiques

ASK Super Regenerative Receiver

ST-RX02-ASK Receiver

315/434 MHz ASK RECEIVER

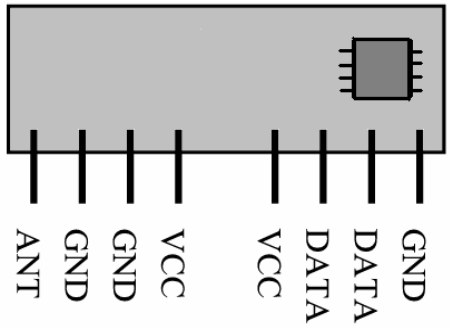
General Description:

The ST-RX02-ASK is an ASK Hybrid receiver module.
A effective low cost solution for using at 315/433.92 MHZ.
The circuit shape of ST-RX02-ASK is L/C.
Receiver Frequency: 315 / 433.92 MHZ
Typical sensitivity: -105dBm
Supply Current: 3.5mA
IF Frequency:1MHz

Features:

- Low power consumption.
- Easy for application.
- Operation temperature range : －20℃～+70℃
- Operation voltage: 5 Volts.
- Available frequency at: 315/434 MHz

Pin Description:



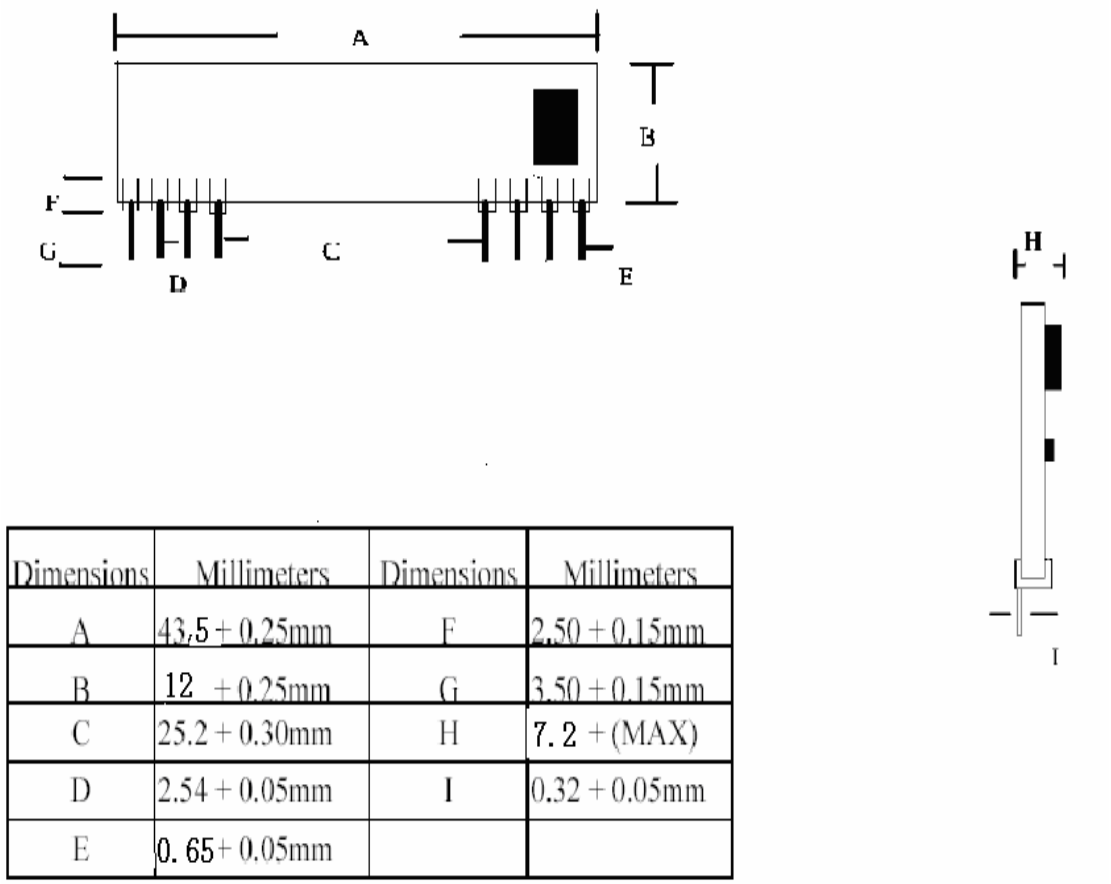
Applications

- Car security system
- Wireless security systems
- Sensor reporting
- Automation system
- Remote Keyless entry

Electrical Characteristics :

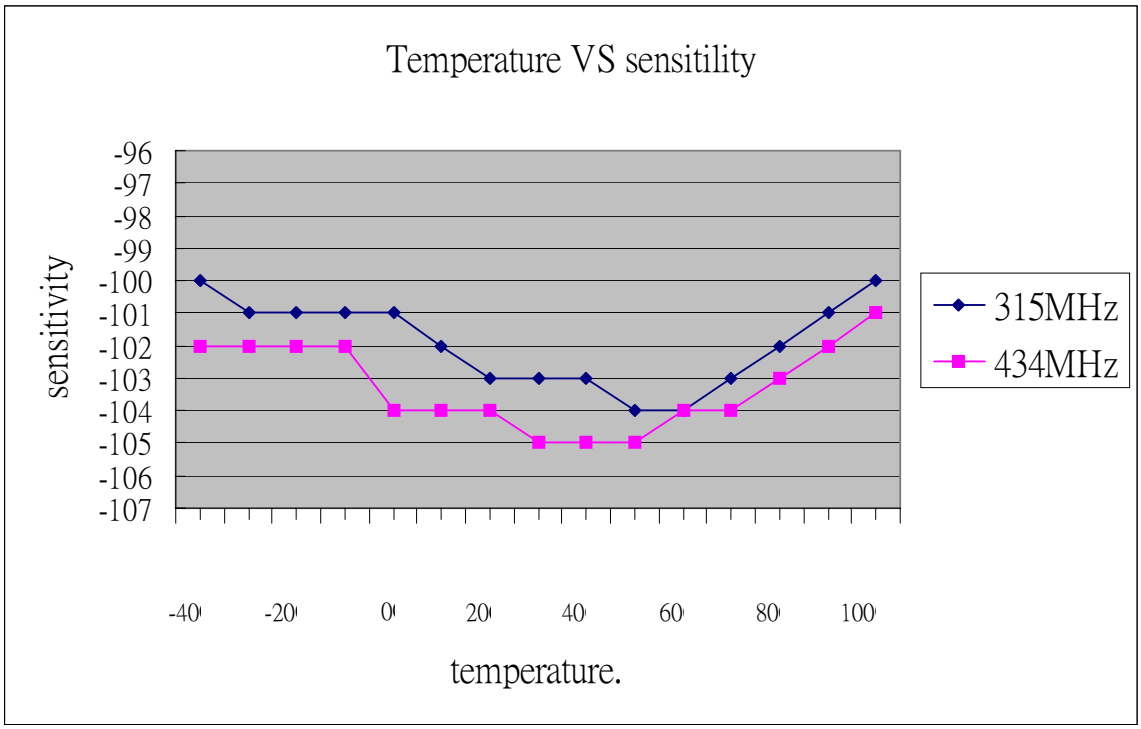
CHARACTERISTIC	MIN	TYP	MAX	UNIT
Vcc Supply Voltage		5		VDC
Is Supply Current		3.5	4.5	mA
FR Receiver Frequency		315/434		MHz
RF Sensitivity(Vcc=5V 1Kbps Data Rate)		-105		dBm
Max Data Rate	300	1k	3k	Kbit/s
VOH High Level Output (I=30uA)	0.7Vcc			VDC
VOL Low Level Output (I=30uA)			0.3Vcc	VDC
Turn On Time(Vcc off-Turn on)		25		ms
TOP Operating Temperature Range	-20		70	℃
Output Duty	40		60	%

Mechanical Dimension:

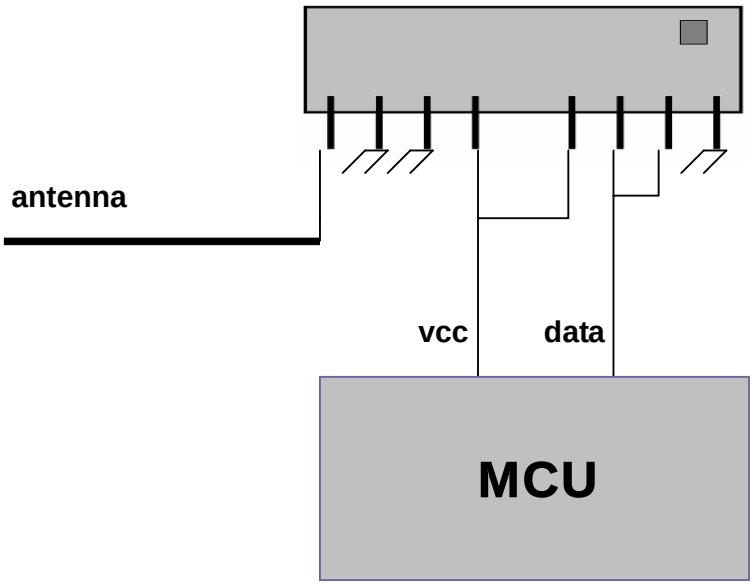


Dimensions	Millimeters	Dimensions	Millimeters
A	43.5 + 0.25mm	F	2.50 + 0.15mm
B	12 + 0.25mm	G	3.50 + 0.15mm
C	25.2 + 0.30mm	H	7.2 + (MAX)
D	2.54 + 0.05mm	I	0.32 + 0.05mm
E	0.65 + 0.05mm		

Typical Characteristics



Typical Application:



Remark:

1. Antenna length about :23cm for 315MHz
17cm for 434MHz

ASK Transmitter Module

ST-TX01-ASK(Saw Type)

General Description:

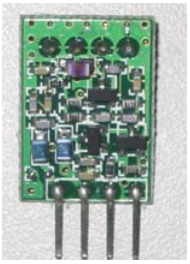
The ST-TX01-ASK is an ASK Hybrid transmitter module. ST-TX01-ASK is designed by the Saw Resonator, with an effective low cost, small size, and simple-to-use for designing.

Frequency Range:315 / 433.92 MHZ.

Supply Voltage: 3~12V.

Output Power : 4~16dBm

Circuit Shape: Saw



315/434 MHz ASK TRANSMITTER

Applications

*Wireless security systems

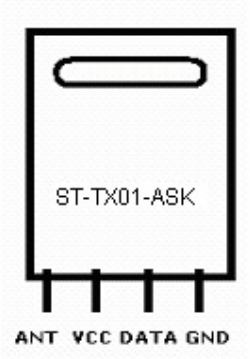
*Car Alarm systems

*Remote controls.

*Sensor reporting

*Automation systems

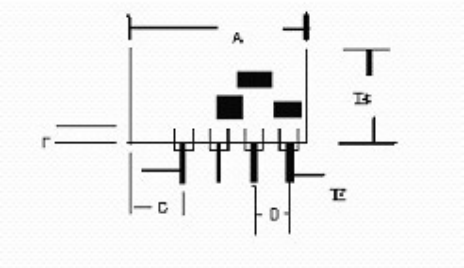
PIN Description:

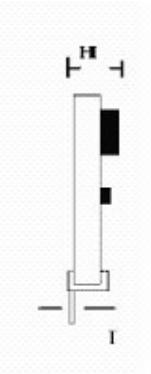


Absolute Maximum Ratings

Parameter	Symbol	Condition		Specification				Unit
				Min.	Typical		Max.	
Operation Voltage					3V	5V	12V		V
Output power	Psens	DATA 5V 1Kbps Data Rate	315MHz		4	10	16		dBm
			Supply current		11	20	57		mA
			434MHz		4	10	16		dBm
			Supply current		11	22	59		mA
Tune on Time	Ton	Data start out by Vcc turn on		10	20				ms
Data Rate				200	1k			3k	bps
Input duty		Vcc=5V; 1kbps data rate		40				60	%
Temperature				-20				+80	℃

Pin Dimension





Dimensions	Millimeters	Dimensions	Millimeters
A	14+0.25mm	F	2.50+0.15mm
B	21+0.25mm	G	3.50+0.15mm
C	4.1+0.30mm	H	5.5mm
D	2.54+0.05mm	I	0.32+0.05mm
E	0.65+0.05mm		

Typical Application: