



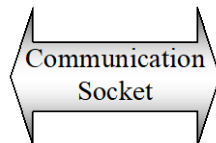
Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

Projet Maison domotique

Programmation de sockets



PC client :
Navigateur,
HTML et
JavaScript



Maison domotique :
Serveur embarqué
programmé en C

© 2021 BFH-TI / E. Firouzi

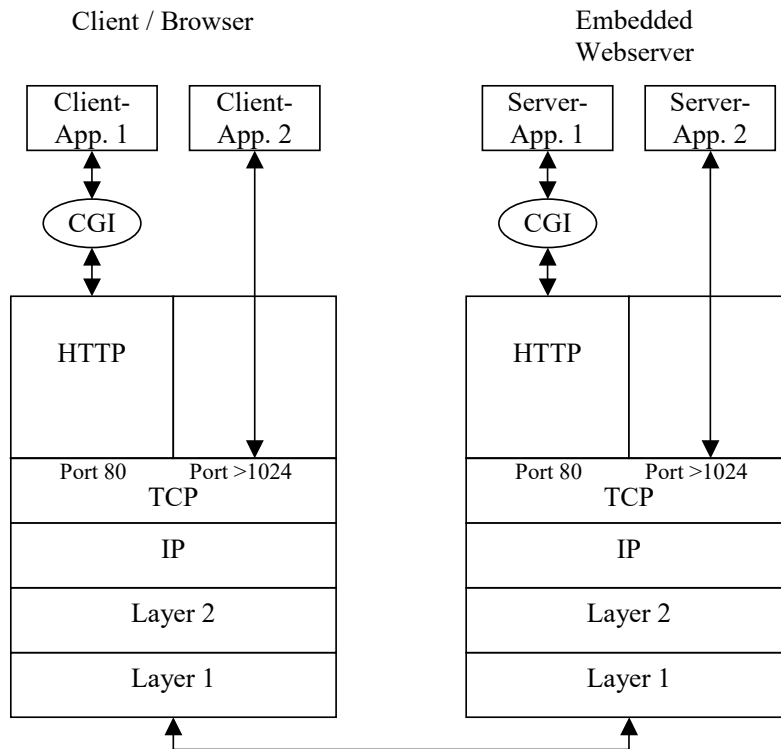
Dernière modification: Décembre 2021
Auteur: Elham Firouzi
Version: 1.1

Table des matières

1	But du projet	3
1.1	Condition cadre	3
1.2	Définition des tâches	3
2	Définition de l'Interface Maison domotique en HTML	4
2.1	Finalité	4
3	Programmation du Serveur Maison domotique en C	5
4	Code de démarrage Linux	7
4.1	Script pour démarrer l'application Maison domotique.....	7
4.2	Srcript pour envoyer un courriel.....	7
5	Installation de VNC Viewer.....	11
5.1	Installation de VNC Viewer sur PC	11
5.2	Activation de VNC Viewer sur Raspberry Pi.....	11
5.3	Definition de la résolution de l'écran	12
5.4	Démarrage de VNC Viewer	13
6	Bibliothèque Maison domotique « Webhouse . h »	15
6.1	Bibliothèque périphérique « bcm2835 . h ».....	16
6.2	Compilation du projet	16
7	Jansson	18
7.1	Installation de la bibliothèque jansson	18

1 But du projet

Le but de ce projet de programmer une interface d'utilisateur graphique, qui permet de commander et de surveiller plusieurs I/O d'un système embarqué. Dans ce contexte, un PC (avec le navigateur pour afficher la page HTML) doit pouvoir se connecter à un système embarqué à l'aide de l'interface Ethernet. Le système embarqué est de type Raspberry Pi et modélise une maison domotique. Les couches pour la communication client - serveur sont représentées dans la figure suivante.



1.1 Condition cadre

- Ce projet sera évalué et fixera une partie de la note du module.
- Le travail doit être réalisé par groupe de 2 étudiants au plus.

1.2 Définition des tâches

Définissez côté client, l'interface d'utilisateur graphique HTML pour envoyer les commandes à la maison domotique ; et côté serveur, le programme en C pour exécuter ces commandes sur la maison domotique. L'Interface d'utilisateur graphique HTML doit permettre d'enclencher ou de déclencher toutes les lampes, le téléviseur, le chauffage et l'alarme. Elle doit également afficher la température ambiante de la maison domotique. En option on devrait pouvoir réguler l'intensité de la lumière ou fixer la température ambiante dans la maison.

2 Définition de l'Interface Maison domotique en HTML

Définissez la page HTML contenant le programme JavaScript pour commander la maison domotique à partir d'un navigateur Internet.

Quelques conseils :

- Définissez un protocole pour la communication entre l'application JavaScript et le serveur embarqué. Idéalement il faudrait transmettre un string donné pour chaque commande. Vous pouvez également afficher ce dernier sur l'écran durant la phase de test. Par exemple pour enclencher la lampe 1, vous pouvez définir le string suivant : "**<L1on>**".
- En JavaScript, la connexion avec le serveur peut être établie de la manière suivante :

```
var websocket = new WebSocket ("ws://192.168.1.100:8000").
```

Dans ce contexte, **192.168.1.100** correspond à l'adresse du serveur embarqué et **8000** au numéro du port de l'application Maison domotique. Les événements socket **onopen** ou **onerror** de l'objet **websocket** permettent en outre de tester, si la connexion a pu être établie avec succès ou non. Le chapitre 25.6 du manuscrit du cours fournit de plus amples informations concernant la classe **WebSocket**.
- Les commandes peuvent être envoyées avec la méthode **send()** de l'objet **websocket** de la manière suivante : **websocket.send("<L1on>") ;**
- Les messages reçus peuvent être lus avec l'événement **onmessage** de l'objet **websocket** de la manière suivante :

```
websocket.onmessage = function (message) {  
    var received_msg = message.data;  
}
```
- Il est fortement recommandé de programmer la première version de la commande domotique avec de simples éléments d'entrées. Par la suite, vous pouvez modifier cette première version en lui intégrant des images ou des graphiques.

2.1 Finalité

L'interface d'utilisateur graphique de la commande de la maison domotique doit pouvoir être téléchargée en introduisant l'adresse IP de cette dernière dans le navigateur Internet. Pour cela, il faut installer dans un premier temps un serveur Internet sur votre kit Raspberry Pi. Les commandes suivantes permettent par exemple d'installer le serveur Internet Apache dans un terminal Shell¹ :

```
sudo apt-get update  
sudo apt-get upgrade  
sudo apt install apache2 -y
```

Ensuite il faut installer tous les documents de votre interface d'utilisateur graphique (HTML, CSS et JavaScript) sur le disque virtuel du serveur Internet. La page HTML principale doit alors être renommée en « **index.html** ». Avec Apache, l'emplacement par défaut du disque virtuel est « **/var/www/html** ». Cependant, cet emplacement peut être modifié dans le fichier de configuration d'Apache « **/etc/apache2/sites-enabled/000-default** ».

¹ <https://pimylifeup.com/raspberry-pi-apache/>

3 Programmation du Serveur Maison domotique en C

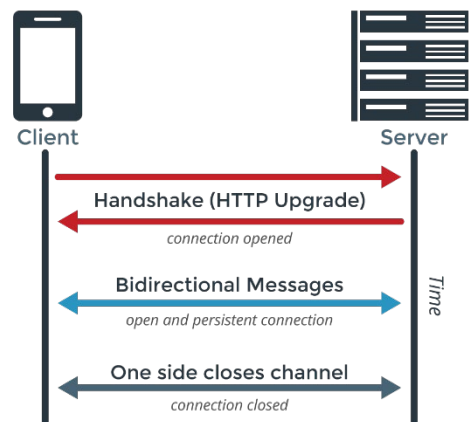
Définissez le programme C pour le côté serveur, qui communique avec la page HTML et qui commande les entrées/sorties de la maison domotique.

Les fonctions des bibliothèque "**Webhouse.h**" et sont mises à disposition pour la commande des entrées/sorties de la maison domotique.

Procédure possible de développement :

- Implémentation de la connexion socket :
 - La connexion socket doit être établie pas pas avec les fonctions des bibliothèques standards `<sys/types.h>` et `<sys/socket.h>`. Ensuite, lorsqu'on arrête l'application, elle doit être fermée correctement. Le chapitre 25.4 du manuscrit fournit une description détaillée de ces fonctions.
 - Attaché le socket serveur au numéro du port **8000** avec la fonction `bind()` du protocole TCP/IP, afin que les utilisateurs puissent établir des connexions au serveur avec ce numéro.
 - Après l'établissement de la connexion au client avec la fonction `connect()`, il faut procéder à une opération de handshake (voir figure ci-dessous). Durant cette procédure, le serveur reçoit une requête handshake avec la fonction `recv()`. Le message de cette requête contient une clé (**Web-Socket-Key**), qui doit être traitée avec l'algorithme de hachage sécurisé 1 (secured hash algorithm 1)². Le résultat de ce traitement (**Web-Socket-Accept**) doit ensuite être renvoyé au client avec la fonction `send()`. La fonction suivante de la bibliothèque "**handshake.h**" est mise à disposition, afin de pouvoir générer la réponse handshake à envoyer en fonction de la requête reçue : `int get_handshake_response(char request[], char response[])`

Le paramètre « **request** » correspond à la requête handshake reçue et le paramètre « **response** » à la réponse à envoyer. **Attention**: l'array « **response** » doit contenir au minimum 130 bytes.



- Après la procédure de handshake, des données peuvent être échangées entre le client et le serveur avec les fonctions `send()` et `recv()`. Afin de pouvoir augmenter la sécurité de transmission, le protocole WebSocket passe par une procédure de cryptage de données³. Afin de pouvoir réaliser ces opérations de cryptage de données, les fonctions suivantes de la bibliothèque "**handshake.h**" ont également été mises à disposition :

```
int decode_incoming_request(char coded_request[], char request[]);
int code_outgoing_response(char respons[], char coded_respons[]);
```

La fonction «**decode_incoming_request**» permet de décoder les requêtes reçues, et la fonction «**code_outgoing_response**» permet de coder les réponses à envoyer. **Attention** : l'array « **request** » doit avoir la même taille que l'array « **coded_respons** », alors que l'array « **coded_response** » doit être deux bytes plus grand de l'array « **respons** ».

L'exemple suivant illustre la procédure à implémenter en C pour traiter les messages reçues :

² https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_servers

³ https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_servers

```

char rxBuf[RX_BUFFER_SIZE];
int rx_data_len = recv (com_sock_id, (void *)rxBuf,
                        RX_BUFFER_SIZE, MSG_DONTWAIT);
// Has a new WebSocket message have been received
if (rx_data_len > 0) {
    rxBuf[rx_data_len] = '\0';
    // Is the message a handshake request
    if (strncmp(rxBuf, "GET", 3) == 0) {
        // Yes -> create the handshake response and send it back
        char response[WS_HS_ACCLLEN];
        get_handshake_response(rxBuf, response);
        send(com_sock_id, (void *)response, strlen(response), 0);
    }
    else {
        /* No -> decode incoming message,
           process the command and
           send back an acknowledge message */
        char command[rx_data_len];
        decode_incoming_request(rxBuf, command);
        command[strlen(command)] = '\0';
        processCommand (command);
        char response[] = "<Command executed>";
        char codedResponse[strlen(response)+2];
        code_outgoing_response(response, codedResponse);
        send(com_sock_id, (void *)codedResponse,
            strlen(codedResponse), 0);
    }
}

```

- Commande des entrées/sorties de la maison domotique :
 - Exploitez les télégrammes, qui sont reçus par l'application JavaScript.
 - Implémentez la fonctionnalité des sorties (lampes, TV, Chauffage).
 - Lisez la température et envoyez cette information au programme JavaScript
 - Implémentez le système d'alarme.

4 Code de démarrage Linux

Linux permet de démarrer des applications directement après la procédure de démarrage. Pour cela il faut modifier le script de démarrage « `/etc/rc.local` ». L'exemple suivant démarre l'application « Maison domotique » à partir de l'instant où le kit Raspberry-Pi possède une adresse IP valide. Dans cet exemple le premier script `rc.local` se contente de démarrer deux autres scripts, à savoir « `emailNotify.sh` » et « `/root/startTemplate.sh` ». Le premier script envoie un courriel contenant l'adresse IP du Kit Raspberry Pi, et le second démarre l'application Maison domotique.

```
#!/bin/sh -e
#
# /etc/rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.
printf "run emailNotify.sh\n"
sudo bash /root/emailNotify.sh &
printf "run startTemplate\n"
sudo bash /root/startTemplate.sh &

exit 0
```

4.1 Script pour démarrer l'application Maison domotique

Le premier script Shell « `/root/tartTemplate.sh` » démarre l'application Maison domotique. Au début de ce script, on attend jusqu'à ce qu'un des interfaces Ethernet ou Wifi soit opérationnel. Ensuite, on démarre l'application Maison domotique en tant que super utilisateur.

```
#!/bin/bash
#
# /root/startTemplate.sh
#
# This script will be called from rc.local
#
# Wait for a valid IP-address
while ! ifconfig | grep "147" > /dev/null &&
      ! ifconfig | grep "192" > /dev/null;
do
    sleep 1
done

# Start the WebSocket server
sudo /home/pi/webhouse/local/Firouzi/Template
exit 0
```

4.2 Sscript pour envoyer un courriel

Le second script Shell « `/root/emailNotify.sh` » envoie un courriel contenant les adresses IP. Au début de ce script, on attend de nouveau jusqu'à ce qu'un des interfaces Ethernet (`eth0`) ou Wifi (`wlan0`) soit opérationnel.

Ensuite on crée un fichier texte contenant les adresses IP des interfaces Ethernet et Wifi. Finalement le contenu du fichier texte est envoyé en tant que courrier électronique avec le script Python **mailing.py**.

```
#!/bin/bash
#
# /root/emailNotify.sh
#
# This script will be called from rc.local

# Wait for a valid IP-address
while ! ifconfig | grep "147" > /dev/null &&
      ! ifconfig | grep "192" > /dev/null;
do
    sleep 1
done

# Define the email message
HOSTNAME='hostname -f'
EIP='hostname -I'
LIP='hostname -i'
echo "$HOSTNAME online" >> /root/email.txt
echo >> /root/email.txt
echo "Ethernet IP address: $EIP" >> /root/email.txt
echo "Local      IP address: $LIP" >> /root/email.txt
echo >> /root/email.txt
date >> /root/email.txt
echo >> /root/email.txt

# Send the email message with mailutils
# mail -s "$HOSTNAME online" elham.firouzi@bfh.ch < /root/email.txt

# Send the email with python
sudo python /root/mailling.py

# Remove the email message
cat /root/email.txt
rm -rf /root/email.txt
exit 0
```

Le script Python « **/root/mailling.py** » envoie électroniquement le fichier texte créé précédemment. Pour cela, il recherche les serveurs d'échange de messagerie parmi les domaines les plus courants. Ensuite, avec chaque serveur trouvé, il envoie un courriel à l'adresse « **elham.firouzi@bfh.ch** » contenant les adresses IP. **Attention** : Le nombre maximal de courriel à envoyer peut être limité avec la variable « **max_successful_email** ».

Afin que vous puissiez recevoir l'adresse IP de votre kit Raspberry Pi par courrier électronique, vous devez remplacer l'adresse courrielle « **elham.firouzi@bfh.ch** » du script « **/root/mailling.py** » par la vôtre.

```
#!/usr/bin/python3
# /root/mailling.py

import subprocess, sys, smtplib, nslookup

from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
```



```
#####
# to change: recipient & max_successful_email
#####
recipient = "elham.firouzi@bfh.ch"
max_successful_email = 3

successful_email = 0

# list of the 21 most popular mail server
# (https://domar.com/pages/smtp_pop3_server)
array_domain = []
array_domain.append('gmail.com')
array_domain.append('live.com')
array_domain.append('office365.com')
array_domain.append('yahoo.com')
array_domain.append('o2.ie')
array_domain.append('ntlworld.com')
array_domain.append('btconnect.com')
array_domain.append('btopenworld.com')
array_domain.append('btinternet.com')
array_domain.append('orange.net')
array_domain.append('wanadoo.co.uk')
array_domain.append('o2online.de')
array_domain.append('t-online.de')
array_domain.append('1and1.com')
array_domain.append('lund1.de')
array_domain.append('comcast.net')
array_domain.append('verizon.net')
array_domain.append('zoho.com')
array_domain.append('mail.com')
array_domain.append('gmx.com')

for domain in array_domain:
    originator = "webhouse@" + domain

    print("\r\nFrom: " + originator)
    print("To:" + recipient)
    print("Domain: " + domain)

    mx_records = []
    mx_values = {'pref' : 0, 'serv' : ''}

    # search every mail exchange server for a given domain
    p = subprocess.Popen('nslookup -type=mx ' + domain + ' 8.8.8.8',
        shell=True, stdout=subprocess.PIPE, stderr=subprocess.STDOUT)
    for line in p.stdout.readlines():
        line = line.decode().lower()
        if line.find("mail exchange") != -1 :
            for char in line:
                if str(char) in "\r\n\t":
                    line = line.replace(char, ' ')
            if line.find("mx preference") != -1 :
                mx_parse = line.replace(' ', '').split(",")
                mx_values['pref'] = int(mx_parse[0].split("=")[1])
                mx_values['serv'] = mx_parse[1].split("=")[1]
            else:
```

```

        mx_parse = line.split(" = ")[1].split(" ")
        mx_values['pref'] = int(mx_parse[0])
        mx_values['serv'] = mx_parse[1]
        mx_records.append(mx_values.copy())
        print("\nline = " + line)

retval = p.wait()

if len(mx_records) == 0 :
    continue

# sort the mail exchange server upon their priority
print("\r\nSearch first priority mail exchange server for \"" +
      domain + "\"")

def mx_pref_sortvalue(record):
    return record['pref']
mx_records=sorted(mx_records, key=mx_pref_sortvalue)

# take the mail exchange server with the highest priority
server = mx_records[0]['serv']

# generate the mail message to send
msg = MIMEMultipart()
msg['From'] = originator
msg['To'] = recipient
msg['Subject'] = "IP-Address Webhouse"

body = open('/root/email.txt', 'r').read()
msg.attach(MIMEText(body, 'plain'))

# send the email with the mail exchange server
print("\r\nSending mail to: \"" + recipient + "\" via server: \"" +
      server + "\"")

try:
    smtp_send = smtplib.SMTP(server, 25)
    try:
        smtp_send.sendmail(originator, recipient, msg.as_string())
        successful_email += 1
    except:
        print("\r\nSending mail to: \"" + recipient +
              "\" via server: \"" + server + "\" has failed!")
    finally:
        smtp_send.quit()

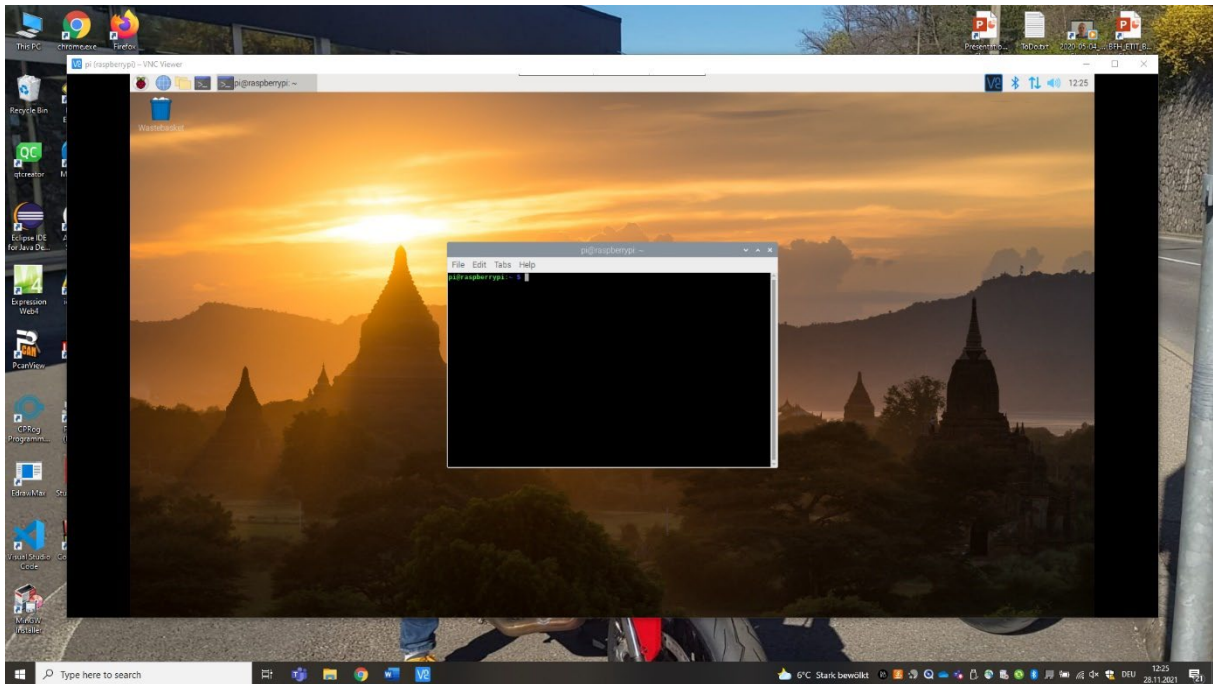
except:
    print("\r\nConnection to the server \"" + server +
          "\" has failed")

if successful_email == max_successful_email:
    sys.exit(0);

```

5 Installation de VNC Viewer

VNC Viewer est un outil logiciel qui permet d'accéder à distance à votre kit Raspberry Pi. Dans ce cas, vous pouvez utiliser votre ordinateur personnel en tant qu'écran, clavier et souris de votre kit Raspberry Pi.

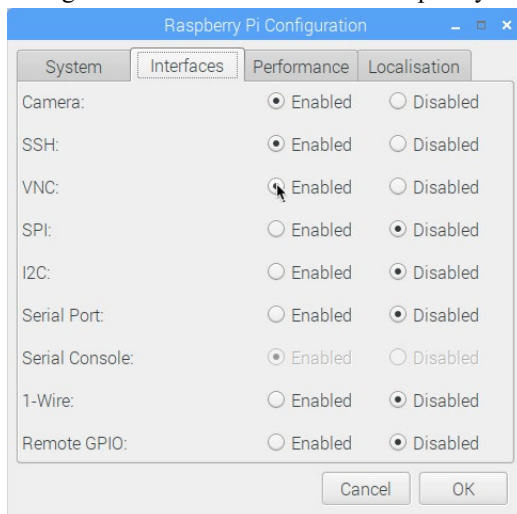


5.1 Installation de VNC Viewer sur PC

L'application « `tightvnc-2.8.63-gpl-setup-64bit.msi` » permet d'installer VNC Viewer sur votre ordinateur personnel. Cette dernière peut être téléchargée soit à partir de ma page Moodle ou à partir du site Internet Officiel de VNC⁴.

5.2 Activation de VNC Viewer sur Raspberry Pi

Afin de pouvoir utiliser VNC Viewer côté Raspberry Pi, il faut activer l'option VNC dans la fenêtre de configuration des interfaces du kit Raspberry Pi⁵.



⁴ <https://www.realvnc.com/en/connect/download/viewer/>

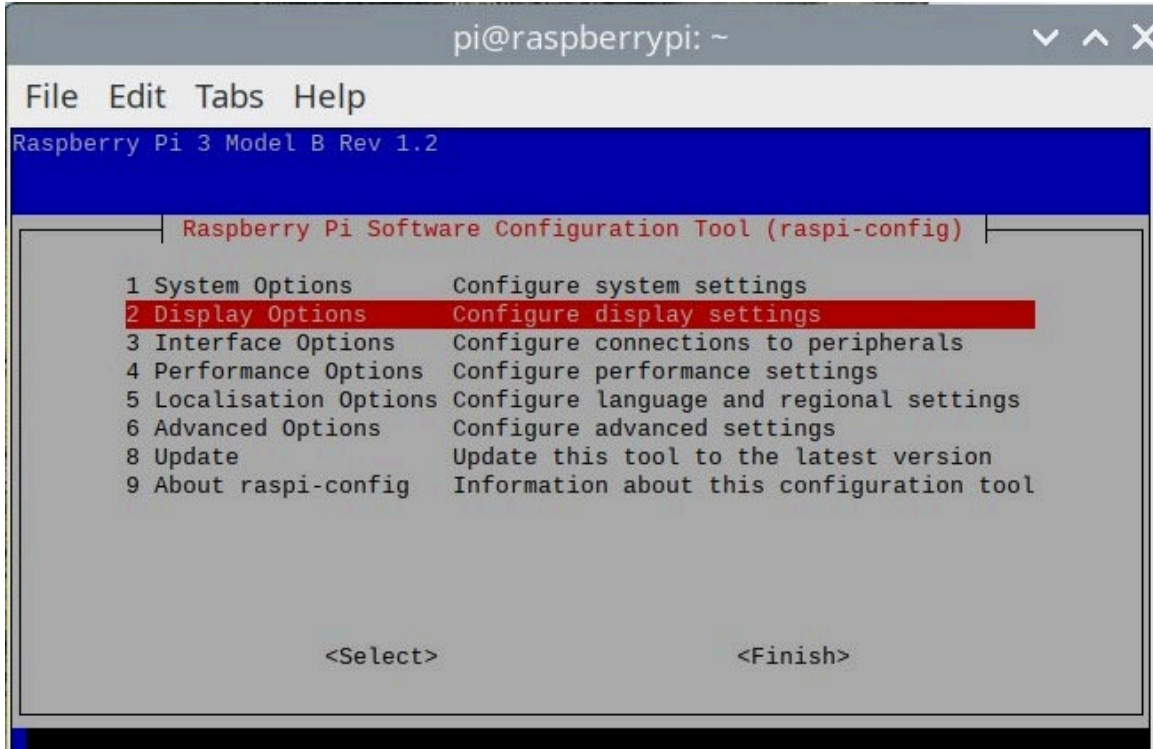
⁵ <https://magpi.raspberrypi.com/articles/vnc-raspberry-pi>

5.3 Definition de la résolution de l'écran pour le mode VNC Viewer

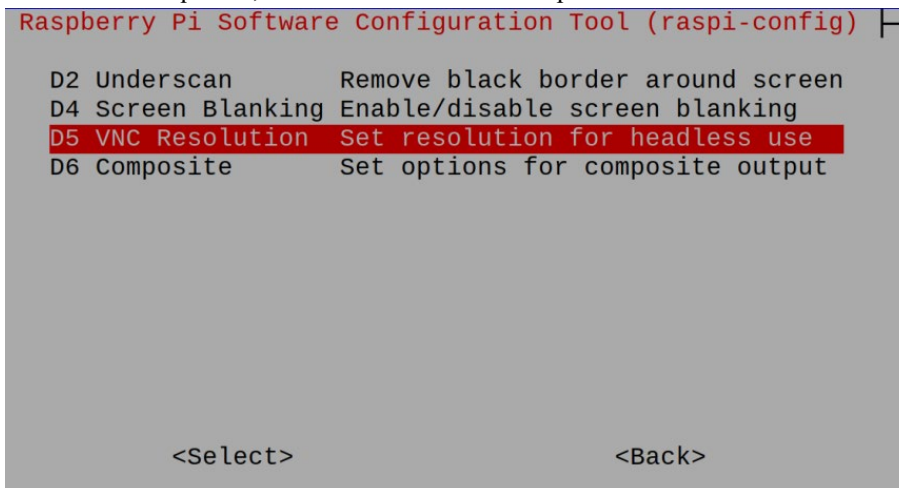
Une fois que vous avez activé l'option VNC Viewer sur votre kit Raspberry Pi, il faut encore fixer la résolution de l'écran pour ce mode sur votre kit Raspberry Pi. Pour cela vous devez introduire la commande suivante dans un shell.

```
sudo raspi-config
```

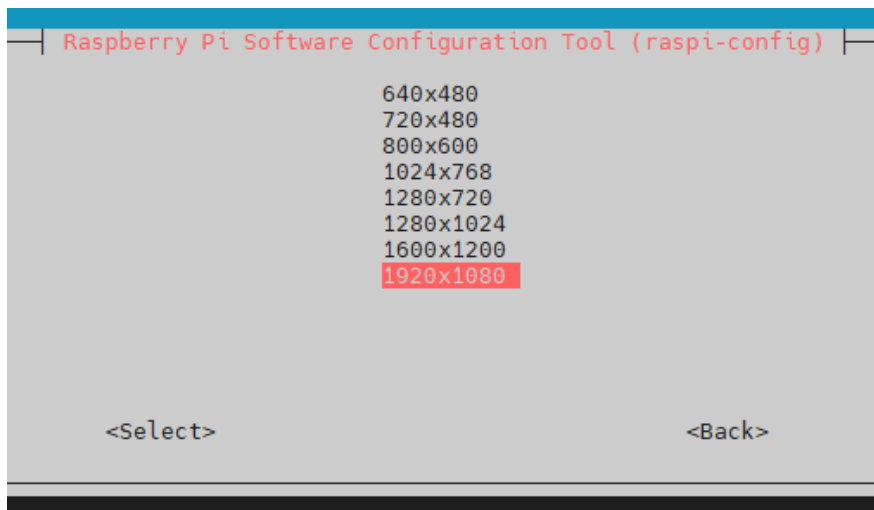
Dans la fenêtre de configuration principale vous devez sélectionner l'option « 2 Display Options ».



Dans la fenêtre qui suit, vous devez sélectionner l'option « D5 VNC Resolution ».



Finalement dans la dernière fenêtre vous pouvez choisir la résolution de l'écran pour le mode VNC Viewer.

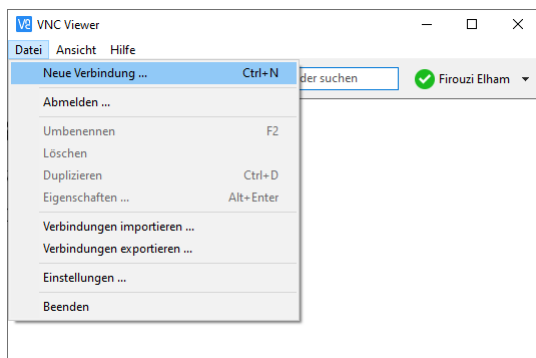


5.4 Démarrage de VNC Viewer

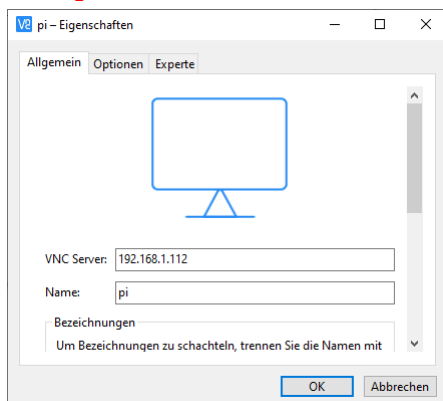
Une fois que l'application VNC Viewer a été complètement installée, vous pouvez la démarrer sur votre ordinateur personnel en cliquant sur son icône.




Après le démarrage de l'application VNC, vous pouvez créer une nouvelle connexion avec le menu « **File/New Connection ...** ».




Durant l'établissement de la connexion, vous devez d'abord introduire l'adresse IP de votre kit Raspberry Pi – que vous avez reçu normalement par courrier électronique – et le nom d'utilisateur « **pi** » dans la fenêtre de dialogue « **Properties** ».



Ensuite, vous devez introduire le nom d'utilisateur (**pi**) ainsi que le mot de passe utilisateur (**pi**), dans la fenêtre de dialogue « **Authentification** ».

 Authentifizierung

✕

 **Authentifizierung bei VNC Server**
192.168.1.112::5900 (TCP)


VNC Server-Anmeldeinformationen eingeben
(Hinweis: NICHT Ihre RealVNC-Kontodaten)

Benutzername:

pi

Kennwort:

••



☐ Kennwort speichern

[Kennwort vergessen?](#)

Schlagwort:

Herman chief explain. Popular economy nice.

Signatur:

f3-8e-cf-d3-4d-01-c9-e5

OK

Abbrechen

6 Bibliothèque Maison domotique « Webhouse.h »

La bibliothèque "**Webhouse.h**" met à disposition les fonctions suivantes pour la commande des entrées/sorties de la maison domotique :

```
//-----Macros-----
#define PWM

#define GPIO_TV          RPI_BPLUS_GPIO_J8_03          //Pin 3
#define GPIO_dimSLamp    RPI_BPLUS_GPIO_J8_32          //Pin 32
#define GPIO_dimRLamp    RPI_BPLUS_GPIO_J8_33          //Pin 33
#define GPIO_Heat        RPI_BPLUS_GPIO_J8_05          //Pin 5
#define GPIO_Alarm        RPI_BPLUS_GPIO_J8_15          //Pin 15
#define GPIO_LED1        RPI_BPLUS_GPIO_J8_07          //Pin 7
#define GPIO_LED2        RPI_BPLUS_GPIO_J8_11          //Pin 11

//-----Function prototypes-----
extern void initWebhouse(void);
extern void closeWebhouse(void);

extern void turnTVOn(void);
extern void turnTVOff(void);
extern int getTVState(void);

extern void dimRLamp(uint16_t Duty_cycle);
extern void dimSLamp(uint16_t Duty_cycle);

extern void turnLED1On(void);
extern void turnLED1Off(void);
extern int getLED1State(void);

extern void turnLED2On(void);
extern void turnLED2Off(void);
extern int getLED2State(void);

extern void turnHeatOn(void);
extern void turnHeatOff(void);
extern int getHeatState(void);
extern float getTemp(void);

extern int getAlarmState(void);
```

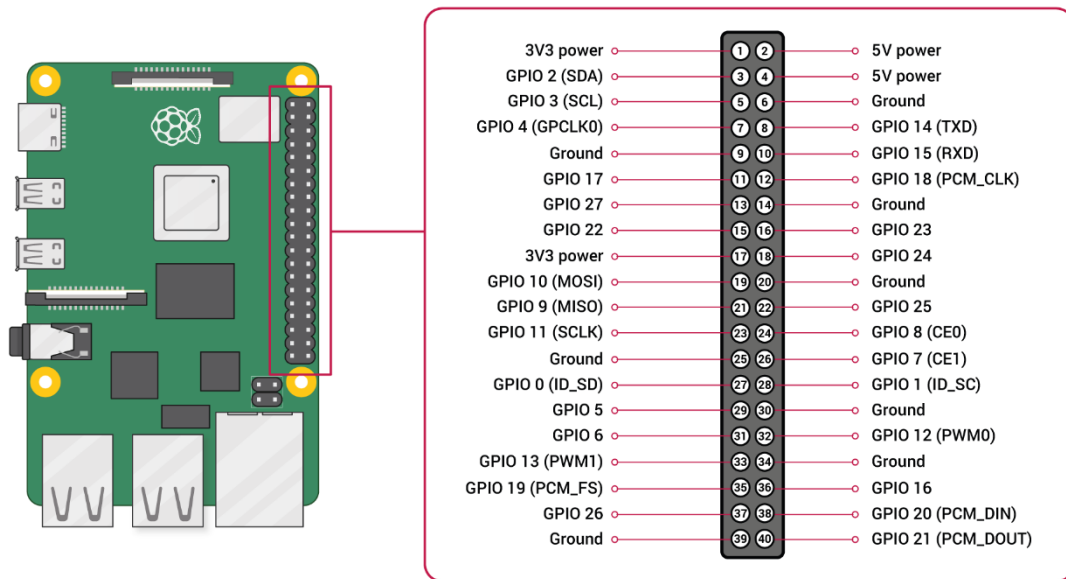
Les fonctions « **initWebhouse()** » et « **closeWebhouse()** » permettent d'initialiser et de clôturer l'environnement de contrôle de la maison domotique. Par conséquent, ces fonctions doivent être appelées resp. tout au début et tout à la fin du programme C. Un fois que l'environnement de contrôle a pu être initialisé avec succès, les autres fonction de contrôle peuvent être appelées à n'importe quel instant.

La macro « **#define PWM** » permet de spécifier l'utilisation de l'interface PWM pour réguler l'intensité des deux lampes. **Attention** : Lorsque le programme est compilé avec cette macro, il faut systématiquement le démarrer en mode super utilisateur de la manière suivante :

sudo ./Template

En effet, l'interface PWM ne fonction qu'en mode super utilisateur.

Les autres macros indiquent l'utilisation des pins d'entrées et de sorties sur le kit Raspberry Pi. La numérotation de ces pins correspond au modèle « Raspberry Pi 3 model B ».



6.1 Bibliothèque périphérique « bcm2835.h »

La bibliothèque "[Webhouse.h](#)" se base sur les fonctions API de la bibliothèque périphérique «bcm2835.h». Il est très probable que cette bibliothèque ne soit pas encore disponible sur votre kit Raspberry Pi. Dans ce cas, vous devez d'abord choisir un dossier temporaire. Ensuite, vous pouvez installer la bibliothèque **bcm2835.h** dans ce dossier à l'aide des commandes de ligne suivantes :

```
wget http://www.airspayce.com/mikem/bcm2835/bcm2835-1.68.tar.gz
tar xvfz bcm2835-1.68.tar.gz
cd bcm2835-1.58
./configure
make
sudo make install
```

6.2 Compilation du projet

La compilation du projet peut être effectuée à l'aide de la commande linux **make** dans le répertoire du projet. Cette commande lance la compilation en fonction du script de compilation **Makefile**, qui contient les directives suivantes :

```
Template.o Webhouse.o handshake.o
gcc -o Template main.o Webhouse.o handshake.o base64.o sha1.o
-lbcm2835 -lpthread -ljansson -lm

main.o: main.c Webhouse.h handshake.h
gcc -c main.c

Webhouse.o: Webhouse.c Webhouse.h
gcc -c Webhouse.c

handshake.o: handshake.c handshake.h base64.h sha1.h
gcc -c handshake.c

base64.o: base64.c base64.h
gcc -c base64.c

sha1.o: sha1.c sha1.h
```



```
gcc -c sha1.c
```

```
clean:
```

```
rm Template main.o Webhouse.o handshake.o base64.o sha1.o
```

7 Jansson

7.1 Installation de la bibliothèque jansson

Il est fort probable que la bibliothèque « **jansson.h** » ne soit pas encore installé sur votre kit Raspberry Pi. Dans ce cas, vous devez d'abord choisir un dossier temporaire. Ensuite, vous pouvez installer la bibliothèque **jansson.h** à l'intérieur de ce dossier en utilisant les commandes de ligne de commande suivantes :

```
wget https://github.com/akheron/jansson/releases/download/v2.14/jansson-2.14.tar.gz
tar xvfz jansson-2.14.tar.gz
cd jansson-2.14
./configure
make
sudo make install
```