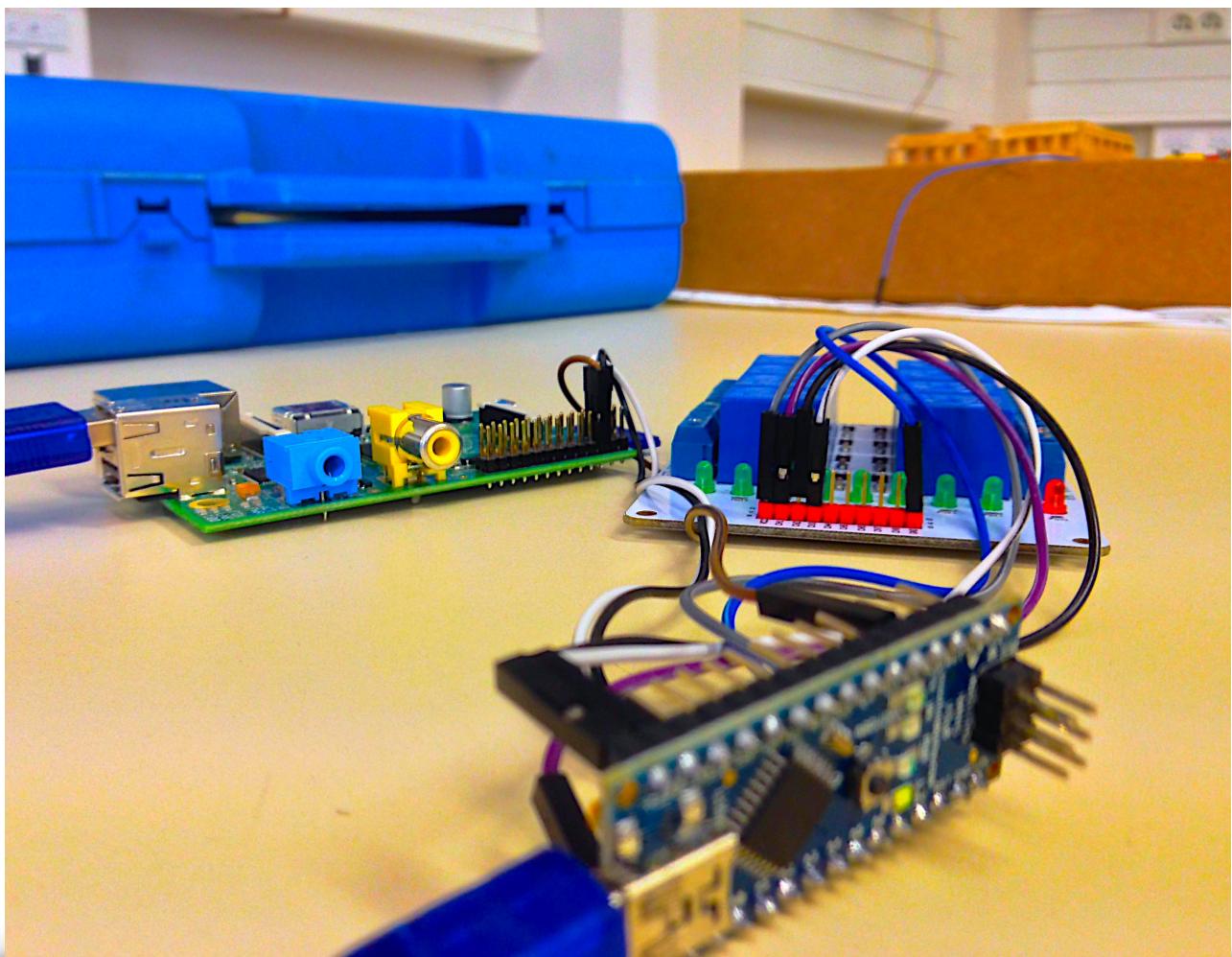


Dossier Projet DomHome

Ruelle Lucas

4 avril 2014

DomHome : Un projet novateur



DomHome : Un projet novateur	1
Sommaire	2
I. Introduction et problématique	1
II. Conception préliminaire	1
1. Cahier des charges	1
2. Solutions possibles	3
3. Choix des solutions	3
4. Modélisation Diagrammes SysML	4
5. Tableau comparatif	6
6. Chaîne d'Information et d'Energie	6
6. Planification du projet grâce à Gant Project	7
III. Conception détaillée	7
1. Choix des composants	7
2. Programmation	8
3. Réalisation des schémas électroniques	9
IV. Maquettage et Prototypage	9
1. Réalisation de la maquette	9
2. Configuration du Système	9
3. Tests et Mesures	10
5. Conclusion et Perspective	10
Définition :	1
Code Source	2
Arduino	2
Script Python	4

I. Introduction et problématique

Le projet DomHome est une initiative prise à la fois pour le projet de Bac STI2D SIN, et à la fois pour innover dans le champ de la domotique et de l'aide à la personne.

Pourquoi la domotique ?

La domotique, c'est le fait de pouvoir contrôler sa maison du bout des doigts grâce à la technologie.

En effet, grâce à cette solution, vous pourrez contrôler l'état de vos appareils électriques simplement en cliquant sur un bouton dans le site web ou dans l'application iOS.

Pourquoi avoir choisi ce projet ?

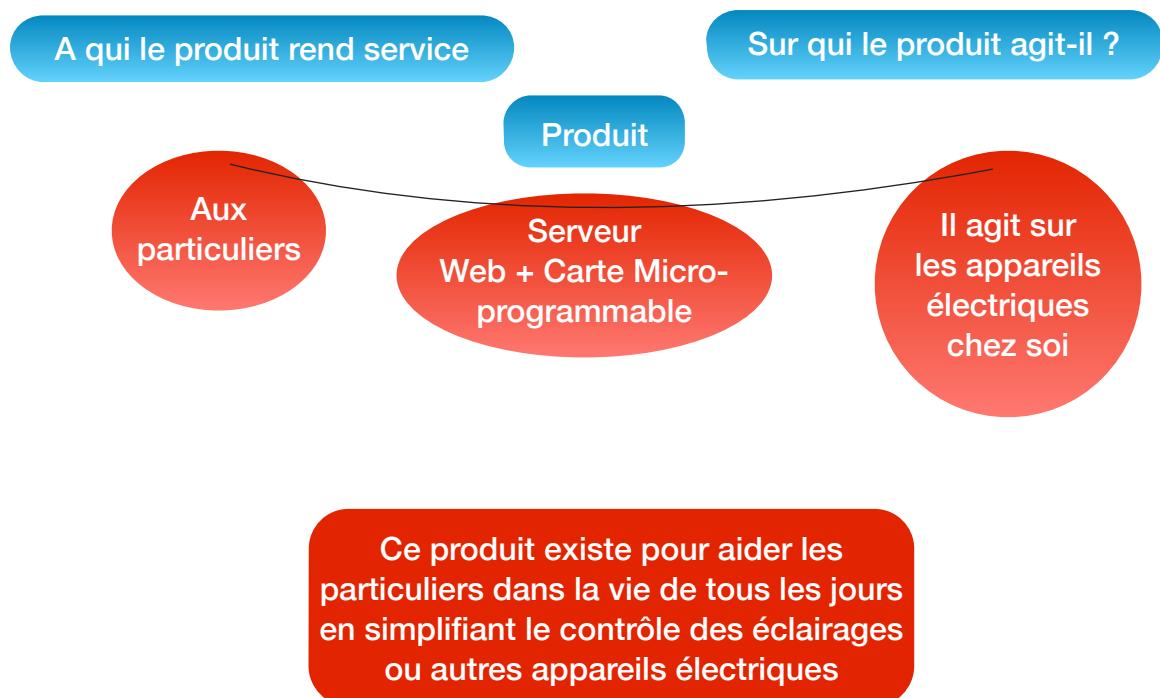
Nous avons choisi ce projet, car le domaine de la domotique a beaucoup de potentiel dû à sa nouveauté et les améliorations possibles presque infinies.

Nous comptions faire une solution libre, que tout le monde pourra re-créer chez soi librement, simplement et sans connaissance spéciale, juste en suivant un tutoriel.

C'est pour ça que nous allons écrire des tutoriels sur des sites comme faitmain.org ou <http://www.instructables.com> qui sont des gros sites DIY (Do it Yourself - Voir Annexe).

II. Conception préliminaire

1. Cahier des charges



Pourquoi ce besoin existe t'il ? (causes, origines...)

Il existe du fait que l'Homme souhaite automatiser presque toutes les tâches manuelles.

Pourquoi ce besoin existe t'il ? (dans quel but, finalités...)

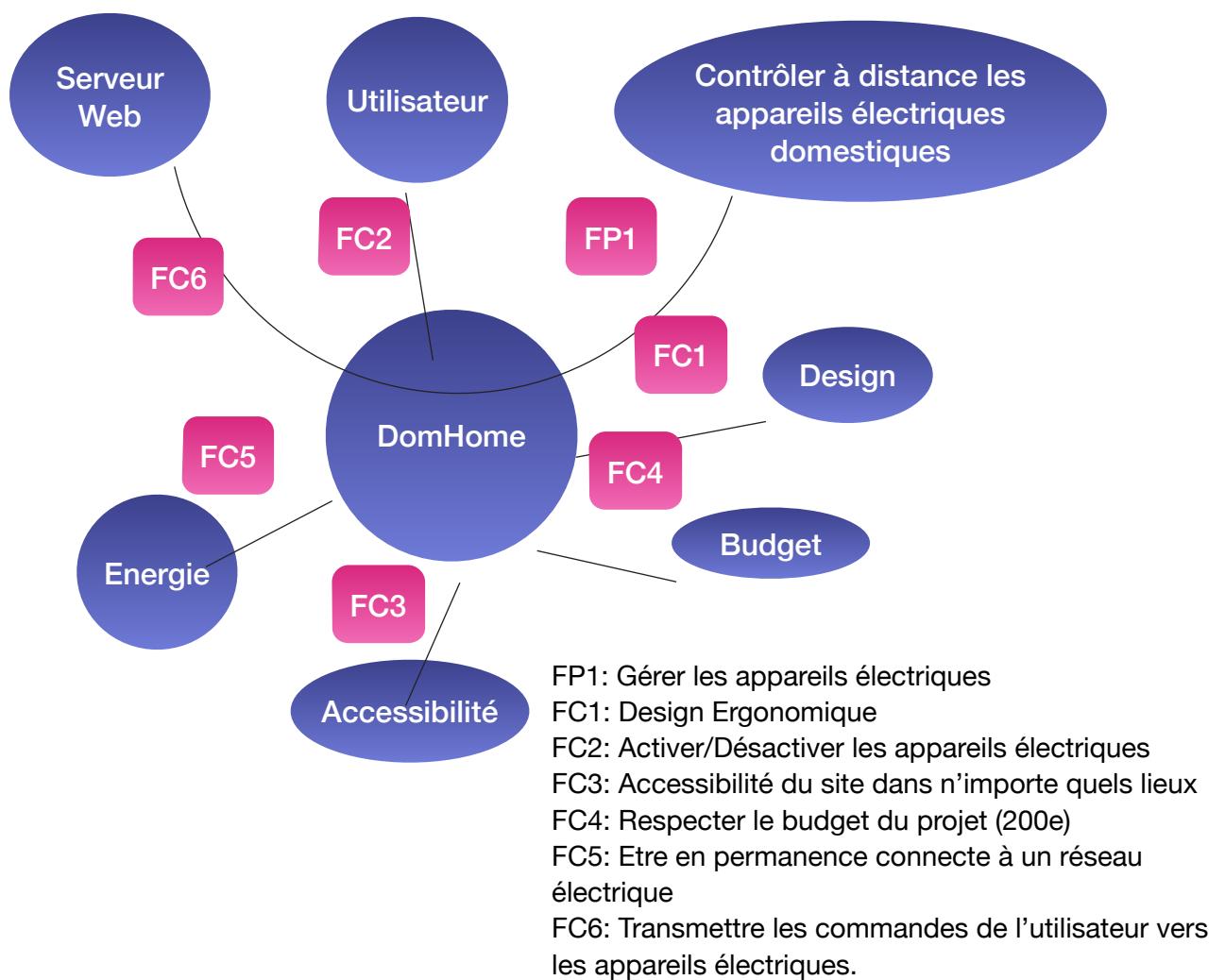
Faciliter la vie de l'Homme, permettre à distance de gérer son habitat, sa consommation d'énergie

Qu'est ce qui pourrait faire évoluer ou disparaître ce besoin ?

La commande par voix pourra faire évoluer ce besoin et le transformer ou le faire disparaître

Quelle est la probabilité de l'évolution ou de la disparition du besoin ?

La technologie de nos jours évolue rapidement, ce qui nous laisse présager qu'il est fort probable qu'il y ait une évolution de ce besoin



2. Solutions possibles

Matérielle:

Plusieurs choix matériels s'offraient à nous, connaissant déjà le Raspberry Pi et l'Arduino, j'en ai alors parlé à mes camarades qui ont approuvé cette idée.



Logicielle:

Plusieurs choix s'offraient aussi à nous, soit j'utilisais Nginx, Lighttpd, LiteSpeedTech soit Apache comme serveur web.

Je pouvais utiliser Raspbian, Archlinux ou Pidora



Je pouvais utiliser du python, ou n'importe quel autre langage (comme le C).



Je pouvais utiliser FlowCode ou tout programmer moi-même.

3. Choix des solutions



Matérielle:

J'ai choisi d'utiliser un Raspberry Pi (voir annexe), un Arduino et une carte Relais car je préfère utiliser un Arduino qu'un PIC16F, plus compliqué pour moi à programmer.

Je prendrai en charge intégralement la programmation.

Nous avons choisi d'utiliser cette solution plutôt qu'un Raspberry Pi + une carte Relais parce que sans Arduino, tous les ports i2C du Raspberry Pi seront utilisés, aucune évolution ne sera possible.

Avec l'Arduino, des ports GPIO seront libres, une évolution est possible comme l'ajout de capteurs de température, d'humidité, de gaz, etc...



Logicielle:

J'hésitais entre Apache et Lighttpd car ils sont compatibles CGI (Common Gateway Interface), et ne connaissant pas assez Lighttpd, j'ai choisi Apache, que je connais beaucoup plus, grâce à des projets personnels.

J'ai choisi d'utiliser Raspbian, qui est en fait Debian compilé ARM (donc compatible Raspberry pi), qui est un système que je connais bien car c'est le 1 er système d'exploitation UNIX que j'ai utilisé sur Raspberry Pi.



Pidora qui est inspiré de Fedora est aussi très bien mais je ne le connais pas assez pour l'utiliser sur ce projet.

Tout comme Archlinux que je connais mieux que Pidora mais qui nécessite plus de configuration que Raspbian.

J'ai choisi d'utiliser l'IDE Arduino Officiel car je préfère faire le code moi-même pour savoir ce qu'il y a dedans, et FlowCode ne proposait d'utiliser l'I2C dans un seul sens uniquement alors que le projet utilise les 2 (Arduino → Raspberry Pi Raspberry Pi → Arduino)

4. Modélisation Diagrammes SysML

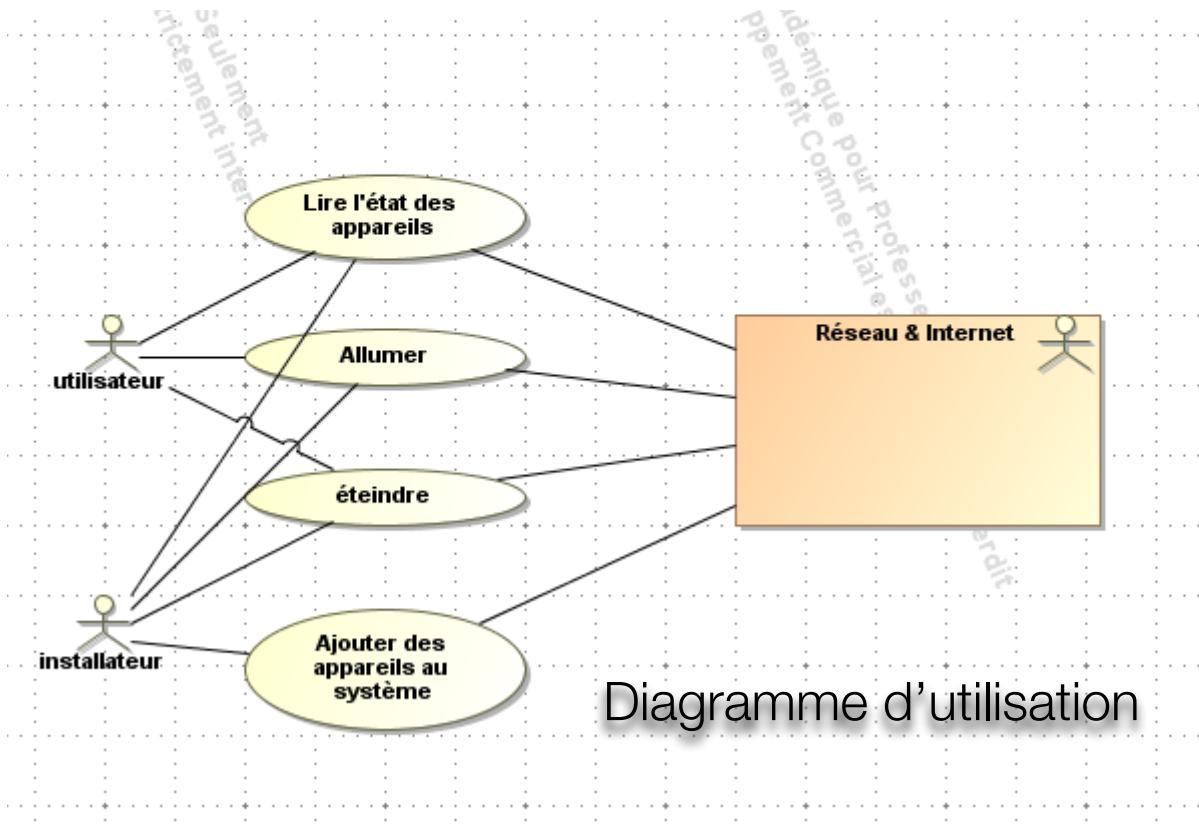


Diagramme d'utilisation

Le diagramme d'utilisation nous indique que l'Installateur/Administrateur possède tous les droits alors que l'utilisateur ne peut qu'intégrer sur l'état des relais ainsi que connaître ce dernier.

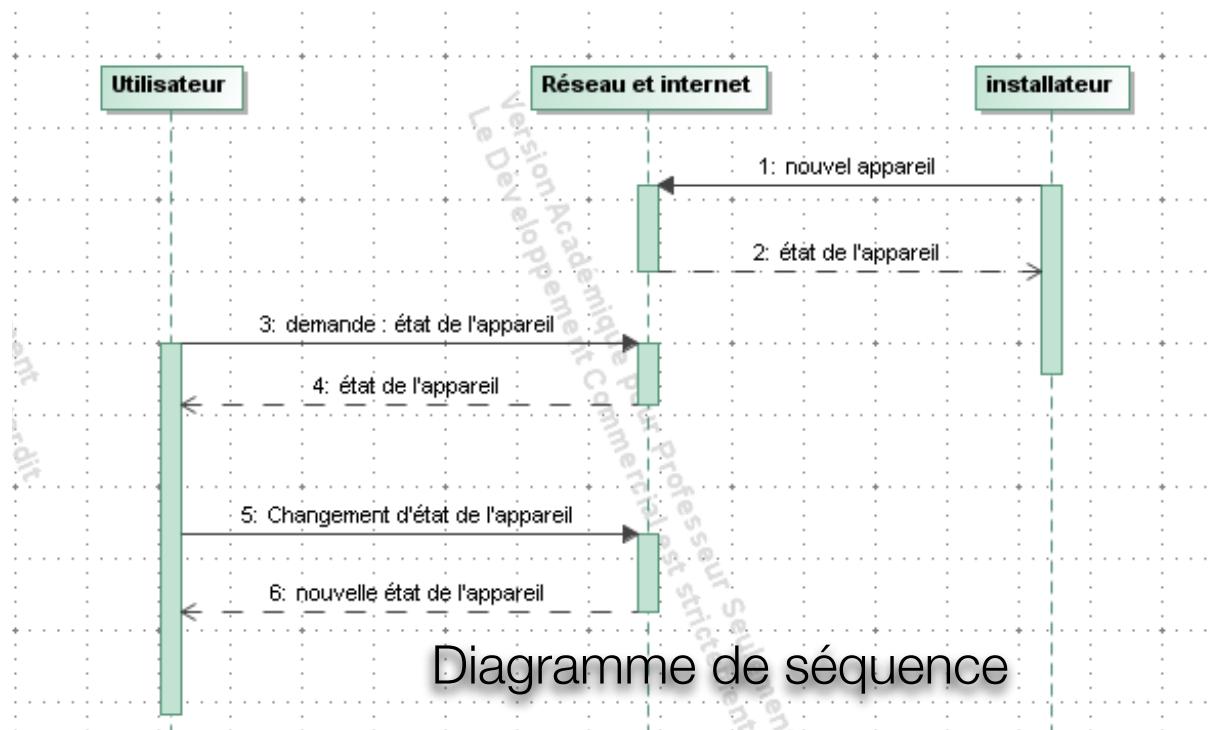


Diagramme de séquence

Le diagramme de séquence nous indique toutes les interactions entre les différents actionneurs et éléments de cette solution domotique.

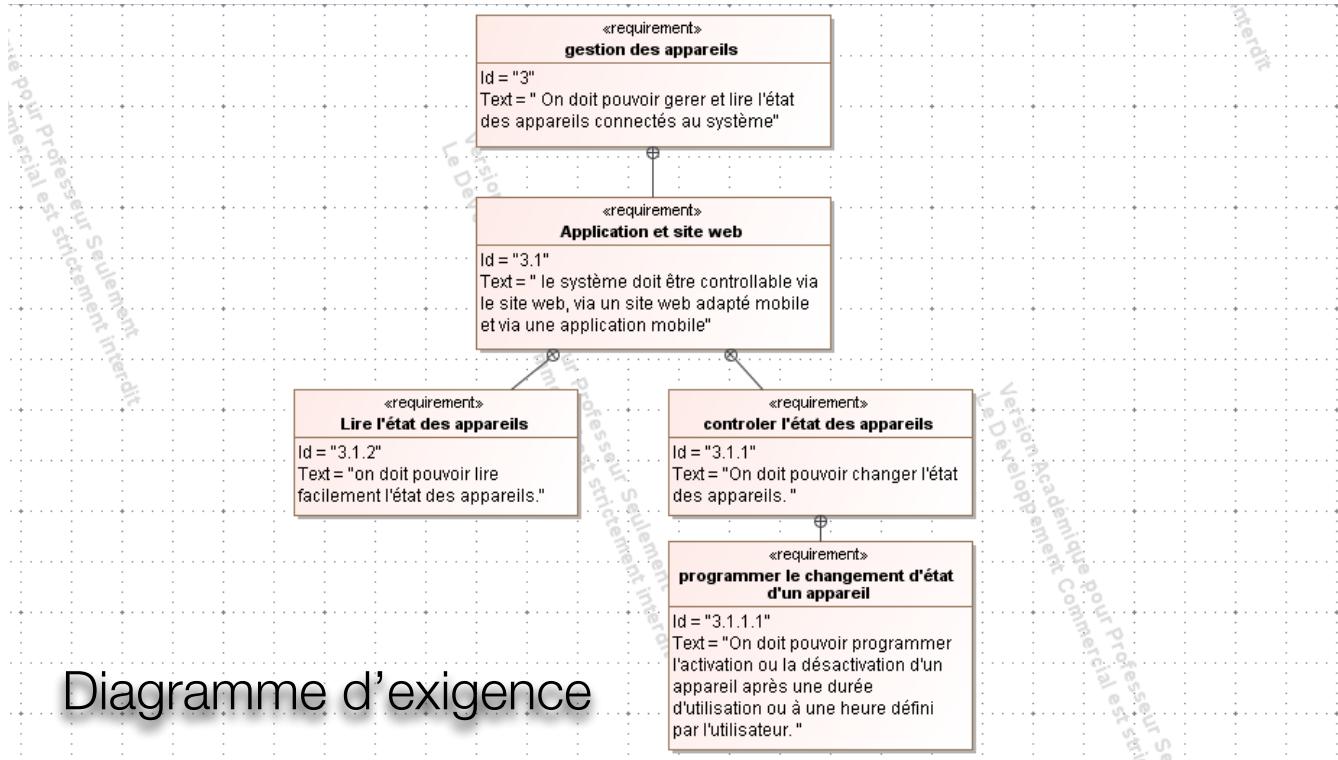


Diagramme d'exigence

Le diagramme d'exigence nous indique sur les exigences que nécessite chaque élément du projet pour son bon fonctionnement.

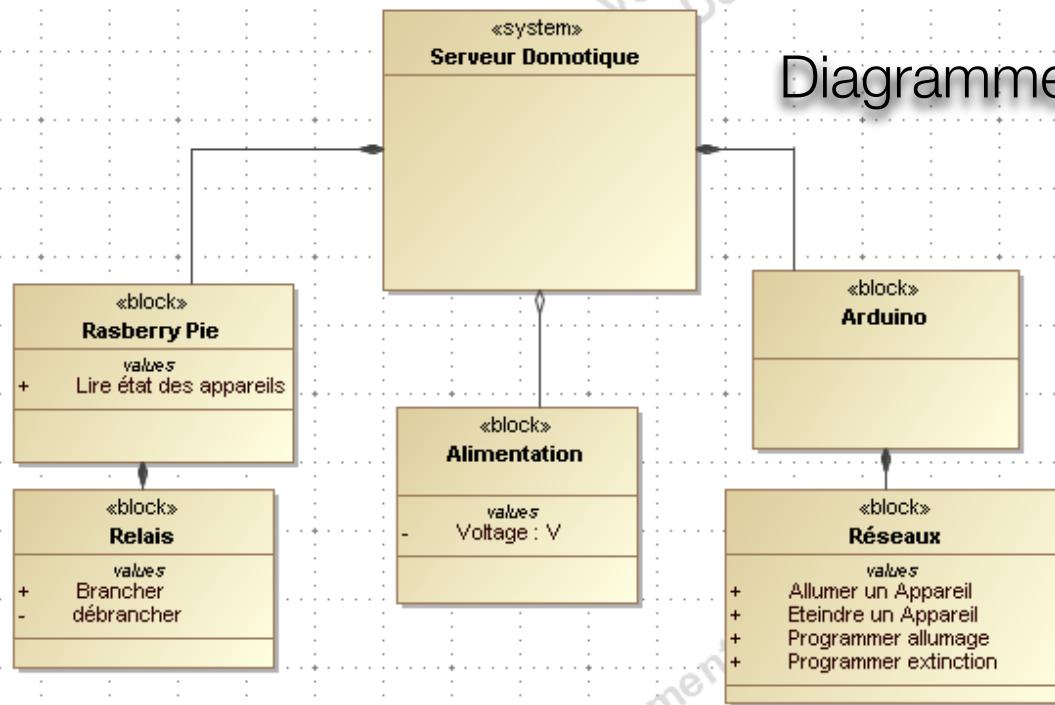
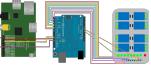


Diagramme de Bloc

Le diagramme de bloc nous indique quant à la nature des différents éléments du projet.

5. Tableau comparatif

	Raspberry Pi + Arduino + Relais		Raspberry Pi + Relais
	Prix : 56,92 euros		Prix : 48,72 euros
Ports i2c de libres = Amélioration avec divers capteurs ou actionneurs		Tous les Ports i2C utilisés	
 Possibilité d'amélioration		 Moins Cher Moins de possibilités de problèmes de communications	
 Plus Cher Moins simple à mettre en place dû aux branchements		 Pas d'amélioration Tous les ports i2c utilisés	

Comme vous pouvez le voir, pour quelques euros supplémentaires, vous pouvez posséder une solution pouvant bénéficier d'améliorations car il y a des ports GPIO de libres.

Comme je l'ai dit en introduction, nous voulions faire une solution open-source, modifiable par tous, et pas bridée, donc si quelqu'un veut la modifier ou ajouter un capteur, c'est possible.

En effet, tout sera disponible sur internet, sur un site web. (notice de montage, code source, tutoriel pour faire le projet à partir des pièces.)

6. Chaîne d'Information et d'Energie

Chaîne d'information

Acquérir
Interface Web

Traiter
Atmega 328P
ARM Processeur
(Liaison i2c)

Communiquer
Ports GPIO

Chaîne d'énergie

Alimenter
5 Volts
2 Ampères

Distribuer
Relais

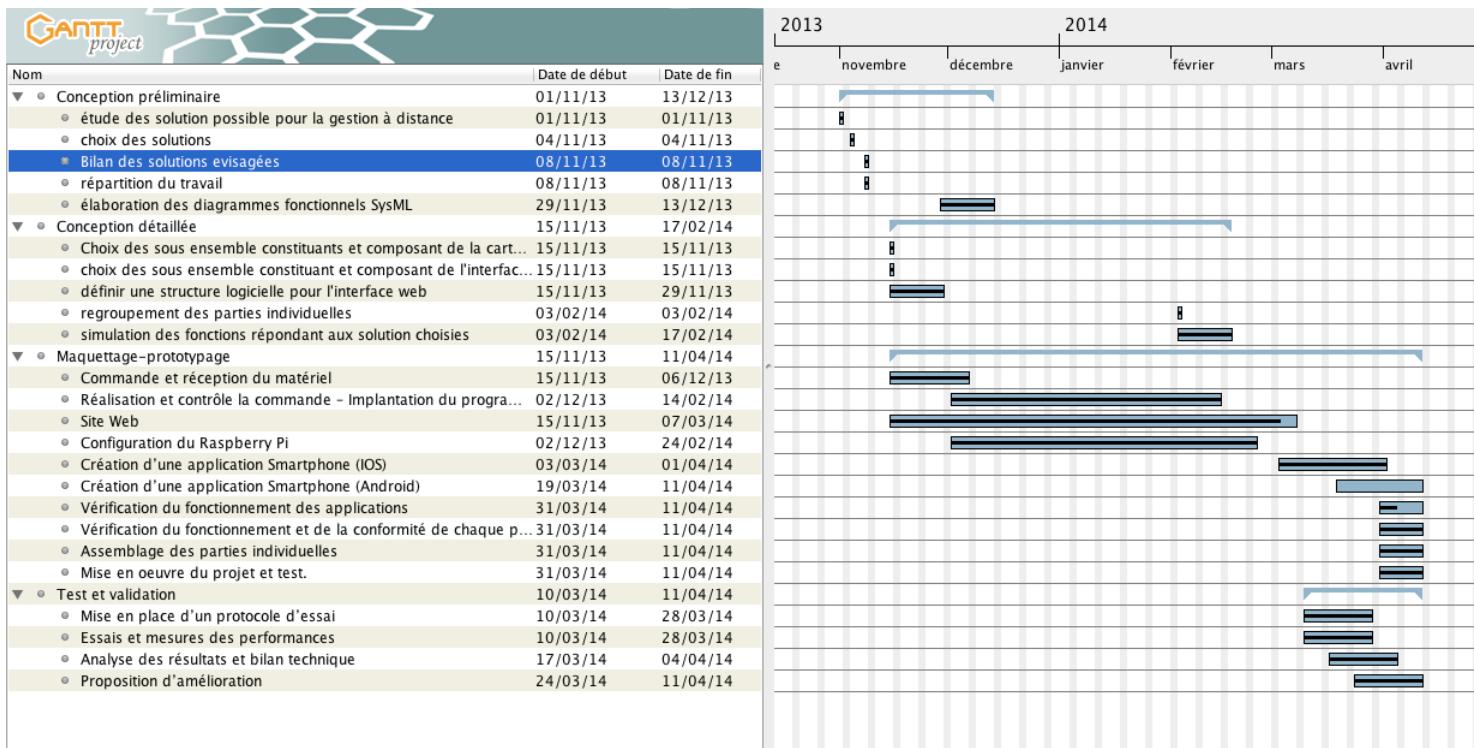
Convertir
Lampe

Transmettre
Environnement



6. Planification du projet grâce à Gant Project

Nous avons établi une planification sommaire sur le Wimi du Projet puis une planification plus avancée et précise grâce à Gantt. (Disponible à l'Annexe 17 en grand)



III. Conception détaillée

1. Choix des composants

Le choix des composants a été rude, il y a plusieurs facteurs à prendre en compte :

- Ecologie
- Consommation d'énergie
- Prix
- Qualité de la pièce
- Compatibilité de la pièce



Nous avons choisi un Raspberry Pi, du fait de sa petite taille pour un ordinateur et sa faible consommation en énergie, ce qui est meilleur pour la planète et l'écologie.

Il est fabriqué par une association anglaise qui prône l'apprentissage de l'informatique à l'école et qui veut diffuser l'informatique plus facilement auprès des enfants avec le Raspberry Pi.

Cet ordinateur vendu seulement 35 euros est donc parfait pour le projet.

La carte micro programmable Arduino est très répandue dans le monde de l'électronique et du DIY (Do It Yourself).

La carte basique officielle ne coûtant que la somme de 20 €, nous avons d'abord choisi la carte appeler Arduino Uno, la plus simple qui nous suffisait largement.

J'ai choisi après mûre réflexion un Arduino Nano non officiel vendu par dealextreme.

Il est plus petit et moins cher que l'officiel, mais son micro-processeur est vide.

J'ai flashé son micro-processeur Atmega 328P moi-même avec le 1 er Arduino UNO.

La carte Relais devait être une carte Relais 5V car la sortie de l'Arduino est 5V ou 3.3V

Nous avons choisi une carte Relais possédant une del sur chaque relais qui indique l'état du relais, cela facilite grandement le développement, car il n'y a pas besoin de brancher de lampes pour tester les programmes.

2. Programmation

La programmation du site web a été effectuée par Romain en PHP. Il utilise aussi des bases SQL pour l'authentification, etc...

Le reste a été fait par moi, Lucas.

J'avais plus de « facilités » que les autres dans la programmation.

J'ai programmé l'Arduino en Arduino (équivalent au C)

J'ai configuré le Raspberry Pi que j'ai mis sous Raspbian (Système d'exploitation).

Le Serveur Web Apache a été configuré pour fonctionner avec la CGI (voir définition pour pourvoir utiliser des scripts python).

J'ai donc fait des scripts python qui affichent une page web qui interagit avec les ports GPIO donc l'Arduino.

L'utilisateur va sur la page web, clique sur le bouton, qui envoie un requête POST au script Python, qui envoie la valeur du POST au port GPIO qui est utilisé pour l'I2C et qui est relié à l'Arduino.

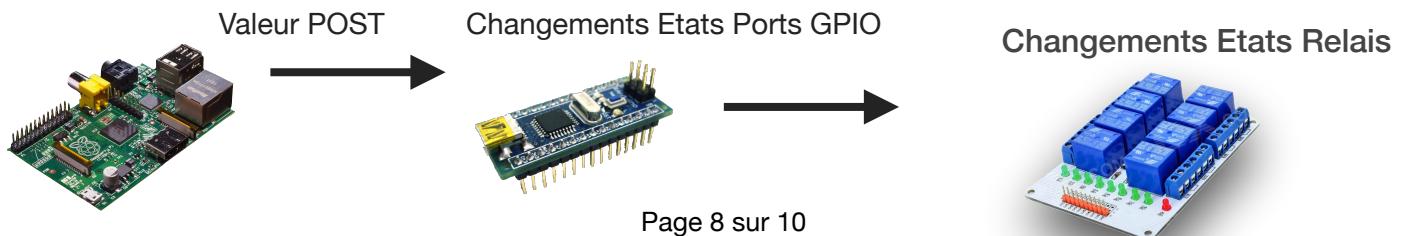
L'Arduino reçoit donc la valeur du POST et l'interprète dans un SWITCH, et en fonction de la valeur, il interagira avec les lampes.

Quand il reçoit le numéro du relais, il va allumer ce relais, et ré-envoyer l'état de ce relais.

S'il reçoit 1, il va allumer le relais 1, et ré-envoyer l'état du relais 1.

Quand il reçoit 100, il ré-envoie l'état de tous les relais.

Github a beaucoup été utilisé pour mettre à jour le code au fur et à mesure du développement... Tous les membres de l'équipe peuvent y accéder



3. Réalisation des schémas électriques

Les schémas électriques sont présents sur les annexes 10 à 16

Nous n'avons pas dessiné les schémas électriques avec ISIS et PROTEUS car ils étaient fournis par les constructeurs, ce sont des Data-Sheets.

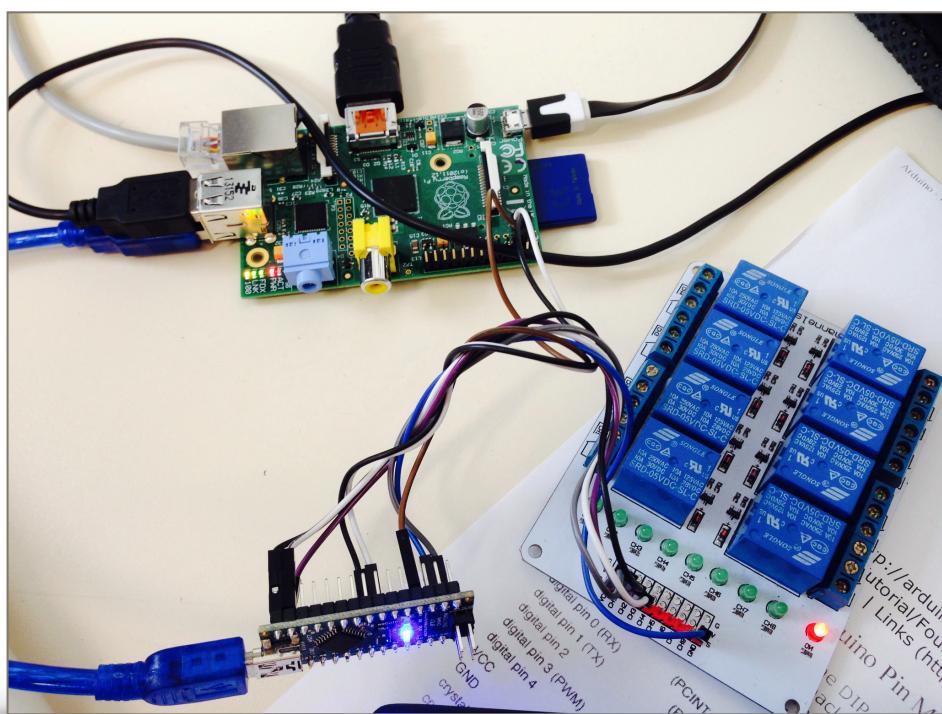
Raspberry Pi : 10 à 14

Carte Relais : 15

Arduino Uno : 16

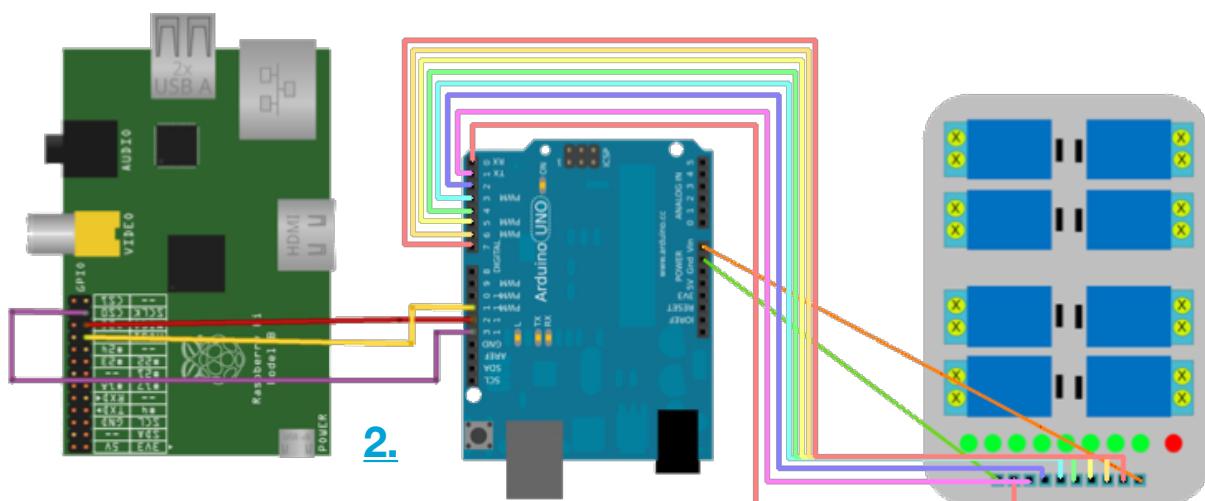
IV. Maquettage et Prototypage

1. Réalisation de la maquette



Comme vous pouvez le voir, les appareils sont reliés entre eux, et il n'y a qu'une source d'alimentation sur le Raspberry Pi.

Ci-dessous, un schéma montrant le projet et les branchements.





Configuration du Système

J'ai effectué la configuration du système en plusieurs parties.

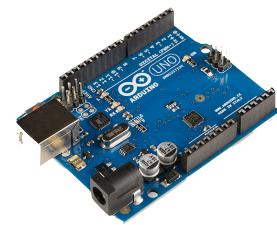
J'ai d'abord configuré l'Arduino pour qu'il puisse recevoir des instructions depuis le Raspberry Pi.

J'ai ensuite configuré le Raspberry Pi pour qu'il puisse envoyer des instructions à l'Arduino à l'aide d'un script python.

J'ai ensuite configuré le serveur Web en CGI pour qu'il puisse afficher le script python rendu compatible dans un navigateur web.

J'ai pu configurer le Raspberry Pi sous Raspbian grâce au SSH.

La page web test a été faite grâce au Twitter BootStrap



3. Tests et Mesures

Le Relais, l'Arduino et le Raspberry Pi sont alimentés avec une seule et unique prise de 5V 2A.

J'ai effectué des tests au fur et à mesure du développement.

Thomas a pratiqué un protocole de test car il fallait aussi des tests du point de vue de quelqu'un d'extérieur au développement.

Il a notamment beaucoup aidé au développement du site web

mais j'ai procédé au test moi-même.



5. Conclusion et Perspective

En conclusion, le développement de notre projet s'est déroulé comme prévu.

J'ai pu tout développer à temps : l'Arduino, le Python/HTML.

J'ai aussi réussi à configurer le serveur Apache pour qu'il soit compatible CGI, ce qui a été le plus difficile.

J'ai été content de participer à ce projet, car j'ai pu développer mes compétences en programmation et en configuration d'UNIX.

Annexe 1

Définition :

Apache :

C'est un des serveurs web les plus répandus grâce à la simplicité de sa configuration, sa stabilité et à son support.

CGI (Common Gateway Interface) :

Méthode d'interaction entre un serveur web et des programmes générateurs de contenu externes, plus souvent appelés programmes CGI ou scripts CGI. Il s'agit de la méthode la plus simple, et la plus courante, pour ajouter du contenu dynamique à votre site web.

Do It Yourself (DIY) :

C'est un mouvement lancé, il y a quelques années, qui prône le fait-soi même.



IDE :

Un IDE est un « environnement de développement intégré ». Il s'agit d'un logiciel proposant des fonctionnalités permettant de fluidifier et de faciliter le développement, que ce soit Web, logiciel, etc.

I2C:

La norme I2C (Inter Integrated Circuit) est une norme développée par Philips en 1982 et actuellement maintenue par NXP (ex-division semiconducteurs de Philips).

Le bus I2C est un bus de données série synchrone bidirectionnel half-duplex.

Plusieurs équipements, soit maîtres, soit esclaves, peuvent être connectés au bus.

Gantt:

C'est un logiciel gratuit et libre de gestion de temps dans un projet .

GPIO (General Port Input/Output):

Ce sont les ports du Raspberry Pi qui sont utilisés pour communiquer avec d'autres périphériques.

SSH:

Secure Shell (SSH) est à la fois un programme informatique et un protocole de communication sécurisé. Le protocole de connexion impose un échange de clés de chiffrement en début de connexion.

Twitter BootStrap:

Twitter Bootstrap est une collection d'outils utile à la création de sites web et applications web. C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs.



Wimi:

Wimi est une solution de productivité tout-en-un qui améliore la collaboration en ligne



Annexe 2

Code Source

Arduino

```
// I2C RELAIS BETA
// by Loiu92 (Lucas Ruelle) <http://loiu92.com>
// Copyright -- Ne pas Redistribuer sans l'Accord de l'Auteur

// Ce programme a été créé par moi pour mon projet de Terminale STI2D SIN, appellé DomHome.
// Ce programme permet de définir l'Arduino en tant que "Slave" I2C du Raspberry PI,
// et de contrôler les sorties de l'Arduino qui sont reliés à une carte Relais.

// Quand l'Arduino reçoit 1, il allume la lampe 1 en laissant passer le courant dans le Relais 1.
// Quand l'Arduino reçoit 11, il éteint la lampe 1 en ne laissant pas passer le courant dans le Relais 1.
// Quand l'Arduino reçoit 2, il allume la lampe 2 en laissant passer le courant dans le Relais 2.
// Quand l'Arduino reçoit 22, il allume la lampe 2 en ne laissant pas passer le courant dans le Relais
// 2.
// Ainsi de suite...

#include <Wire.h>

#define SLAVE_ADDRESS 0x12
int dataReceived = 0;      // Set de la variable dataReceived à 0
int dataResponse=0;        // Set de la variable dataResponse à 0
int relais1 = 13;          // Set de la variable relais1 à 13
int relais2 = 12;          // Set de la variable relais2 à 7
int relais3 = 11;          // Set de la variable relais3 à 8

void setup() {
    Serial.begin(9600);
    Wire.begin(SLAVE_ADDRESS);
    Wire.onReceive(receiveData);
    Wire.onRequest(sendData);
    pinMode(relais1, OUTPUT);    // Initialise la variable relais1 comme pin digital et donc comme
sortie.
    pinMode(relais2, OUTPUT);    // Initialise la variable relais2 comme pin digital et donc comme
sortie.
    pinMode(relais3, OUTPUT);    // Initialise la variable relais3 comme pin digital et donc comme
sortie.

}

void loop() {                // Sous-Programme boucle
    delay(100);              // de 100 ms
}
void receiveData(int byteCount){
    while(Wire.available()) {
        dataReceived = Wire.read();
```

Annexe 3

```
Serial.print(dataReceived);
    relais(dataReceived);
}
}

void EditReponse(int addrRelai){
if (addrRelai!=100){
    dataReponse=digitalRead(addrRelai);
}
else{
    dataReponse=EtatTot();
}
}

void sendData(){
    Serial.print(dataReponse);
    Wire.write(dataReponse);

}

int EtatTot(){
    return digitalRead(relais1)+digitalRead(relais2)*10+digitalRead(relais3)*100;
}

int Etat(int relai){
    int res=digitalRead(relai);
    if (res==1){
        return relai;
    }
    else{
        return relai*11;
    }
}

void relais(int commande){
switch(commande){           // Fonction Appelé Switch qui permet de créer des "conditions".
    case 1:                // Cas 1 (Quand la variable commande est égale à 1)
        digitalWrite(relais1, HIGH); // Ferme le relais 1 et allume la lampe 1
        EditReponse(relais1);
        break;                 // Arret du sous-programme
    case 11:                // Cas 11 (Quand la variable commande est égale à 11)
        digitalWrite(relais1, LOW); // Ferme le relais 1 et allume la lampe 1
        EditReponse(relais1);
        break;                 // Arret du sous-programme
    case 2:
        digitalWrite(relais2, HIGH); // Ferme le relais 1 et allume la lampe 1
        EditReponse(relais2);
        break;
}
```

Annexe 4

```
case 22:  
    digitalWrite(relais2, LOW); // Ferme le relais 1 et allume la lampe 1  
    EditReponse(relais2);  
break;  
case 3:  
    digitalWrite(relais3, HIGH); // Ferme le relais 1 et allume la lampe 1  
    EditReponse(relais3);  
break;  
case 33:  
    digitalWrite(relais3, LOW); // Ferme le relais 1 et allume la lampe 1  
    EditReponse(relais3);  
break;  
case 100:  
    EditReponse(100);  
break;  
  
}  
  
}
```

Script Python

```
#!/usr/bin/env python  
  
import smbus  
import time  
import sys  
import cgi  
import cgitb  
import os  
  
cgitb.enable()  
bus = smbus.SMBus(1) #remplacer le 1 par 0 si vieux raspberry pi  
address = 0x12 #adresse i2c  
form = cgi.FieldStorage()  
bus.write_byte(address, 100)  
#mon_fichier = open("etatrelais.txt", "w")  
#mon_fichier.write(str(bus.read_byte(address)))  
#mon_fichier.close()  
#mon_fichier = open("etatrelais.txt", "r")  
#etatrelais = str(mon_fichier.readlines())  
#etatrelais = str(etatrelais)  
print "Content-type: text/html\n\n"  
  
print """  
<html lang="fr"><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
    <meta charset=<< utf-8 >>
```

Annexe 5

```
<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1">

<meta name="description" content="Site Web de Domhome">
<meta name="author" content="Loiu92">
<link rel="shortcut icon" href="http://getbootstrap.com/assets/ico/favicon.ico">

<title>Theme Template for Bootstrap</title>

<!-- Bootstrap core CSS -->
<link href=files/bootstrap.min.css rel="stylesheet">
<!-- Bootstrap theme -->
<link href=files/bootstrap-theme.min.css rel="stylesheet">

<!-- Custom styles for this template -->
<link href=files/theme.css rel="stylesheet">

<!-- Just for debugging purposes. Don't actually copy this line! -->
<!--[if lt IE 9]><script src="..../assets/js/ie8-responsive-file-warning.js"></script><![endif]-->

<!-- HTML5 shim and Respond.js IE8 support of HTML5 elements and media queries -->
<!--[if lt IE 9]>
  <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
  <script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>
<![endif]-->
<style type="text/css"></style><style id="holderjs-style" type="text/css"></style></head>

<body role="document" style="">

<!-- Fixed navbar -->
<div class="navbar navbar-default navbar-fixed-top" role="navigation">
  <div class="container">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="index.py">DomHome</a>
    </div>
    <div class="navbar-collapse collapse">
      <ul class="nav navbar-nav">
        <li class="active"><a href=index.py>Accueil</a></li>
        <li><a href=apropos.html>A Propos</a></li>
```

Annexe 6

```
<li><a href="http://getbootstrap.com/examples/theme/contact">Contact</a></li>
<li class="dropdown">
    <a href="http://getbootstrap.com/examples/theme/" class="dropdown-toggle" data-
    toggle="dropdown">Dropdown <b class="caret"></b></a>
    <ul class="dropdown-menu">
        <li><a href="http://getbootstrap.com/examples/theme/">Action</a></li>
        <li><a href="http://getbootstrap.com/examples/theme/">Another action</a></li>
        <li><a href="http://getbootstrap.com/examples/theme/">Something else here</a></li>
        <li class="divider"></li>
        <li class="dropdown-header">Nav header</li>
        <li><a href="http://getbootstrap.com/examples/theme/">Separated link</a></li>
        <li><a href="http://getbootstrap.com/examples/theme/">One more separated link</a></li>
    </ul>
</li>
</ul>
</div><!-- .nav-collapse -->
</div>
</div>

<div class="container theme-showcase" role="main">

    <!-- Main jumbotron for a primary marketing message or call to action -->
    <div class="jumbotron">
        <div class="row">
            <div class="col-sm-4">
                <div class="panel panel-default">
                    <div class="panel-heading">
                        <h3 class="panel-title">Relais 1</h3>
                    </div>
                    <div class="panel-body">
                        <form action=index.py method="POST">
                            <button type="submit" name="relais" class="btn btn-sm btn-success"
value="1">Allumer</button>
                            <button type="submit" name="relais" class="btn btn-sm btn-danger"
value="11">Eteindre</button>
                        </div>
                    </div>
                </div>
            <div class="col-sm-4">
                <div class="panel panel-default">
                    <div class="panel-heading">
                        <h3 class="panel-title">Relais 2</h3>
                    </div>
                    <div class="panel-body">
                        <button type="submit" name="relais" class="btn btn-sm btn-success"
value="2">Allumer</button>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
```

Annexe 7

```
<button type="submit" name="relais" class="btn btn-sm btn-danger" value="22">Eteindre</button>
    </div>
    </div>
</div>
<div class="col-sm-4">
    <div class="panel panel-default">
        <div class="panel-heading">
            <h3 class="panel-title">Relais 3</h3>
        </div>
        <div class="panel-body">
            <button type="submit" name="relais" class="btn btn-sm btn-success" value="3">Allumer</button>
            <button type="submit" name="relais" class="btn btn-sm btn-danger" value="33">Eteindre</button>
        </div>
    </div>
</div>
<button type="submit" name="relais" class="btn btn-sm btn-default" value="100">Etat Total</button>
</div>
</div>
"""
if os.environ['REQUEST_METHOD'] == 'POST':
#maintenance    mon_fichier = open("fichier.txt", "w")
#maintenance    mon_fichier.write(str(form))
#maintenance    mon_fichier.close()

    bus.write_byte(address,int(form["relais"].value))
    print "<h1>Etat des Relais :</h1>"
    print "etat relais =" +str(bus.read_byte(address))

#Ecriture Etat relais fichier et page

bus.write_byte(address, 100)
mon_fichier = open("etatrelais.txt", "w")
mon_fichier.write(str(bus.read_byte(address)))
mon_fichier.close()
mon_fichier = open("etatrelais.txt","r")
etatrelais = str(mon_fichier.readlines())

if len(etatrelais)!=7:
    etatrelais=etatrelais[:2] + '0' + etatrelais[2:]
if len(etatrelais)!=7:
    etatrelais=etatrelais[:2] + '0' + etatrelais[2:]
```

Annexe 8

```
mon_fichier.close()
print """
<div align="center">
<TABLE BORDER="1">
  <CAPTION>Etat des relais</CAPTION>
  <TR>
    <TH> Relais 1 </TH>
    <TH> Relais 2 </TH>
    <TH> Relais 3 </TH>
    <TH> Relais 4 </TH>
  </TR>
  <TR>
    <TH>
      print etatrelais[4]
      print """
      </TH>
      <TD>
        """
      print etatrelais[3]
      print """
      </TD>
      <TD>
        """
      print etatrelais[2]
      print """
      </TD>
      <TD> </TD>
    </TR>
  </TABLE>
</div>

<!-- Bootstrap core JavaScript
=====
<!-- Placed at the end of the document so the pages load faster -->
<script src=".templatebootstrap_files/jquery.min.js"></script>
<script src=".templatebootstrap_files/bootstrap.min.js"></script>
<script src=".templatebootstrap_files/docs.min.js"></script>

</body></html>

"""

```

Annexe 9

```
#print etatrelais
#if os.environ['REQUEST_METHOD'] == 'POST':
#maintenance mon_fichier = open("fichier.txt", "w")
#maintenance     mon_fichier.write(str(form))
#maintenance     mon_fichier.close()

#     bus.write_byte(address,int(form["relais"].value))
#     print "<h1>Etat des Relais :</h1>"
#     print "etat relais =" +str(bus.read_byte(address))

#Ecriture Etat relais fichier et page

#bus.write_byte(address, 100)
#mon_fichier = open("etatrelais.txt", "w")
#mon_fichier.write(str(bus.read_byte(address)))
#mon_fichier.close()

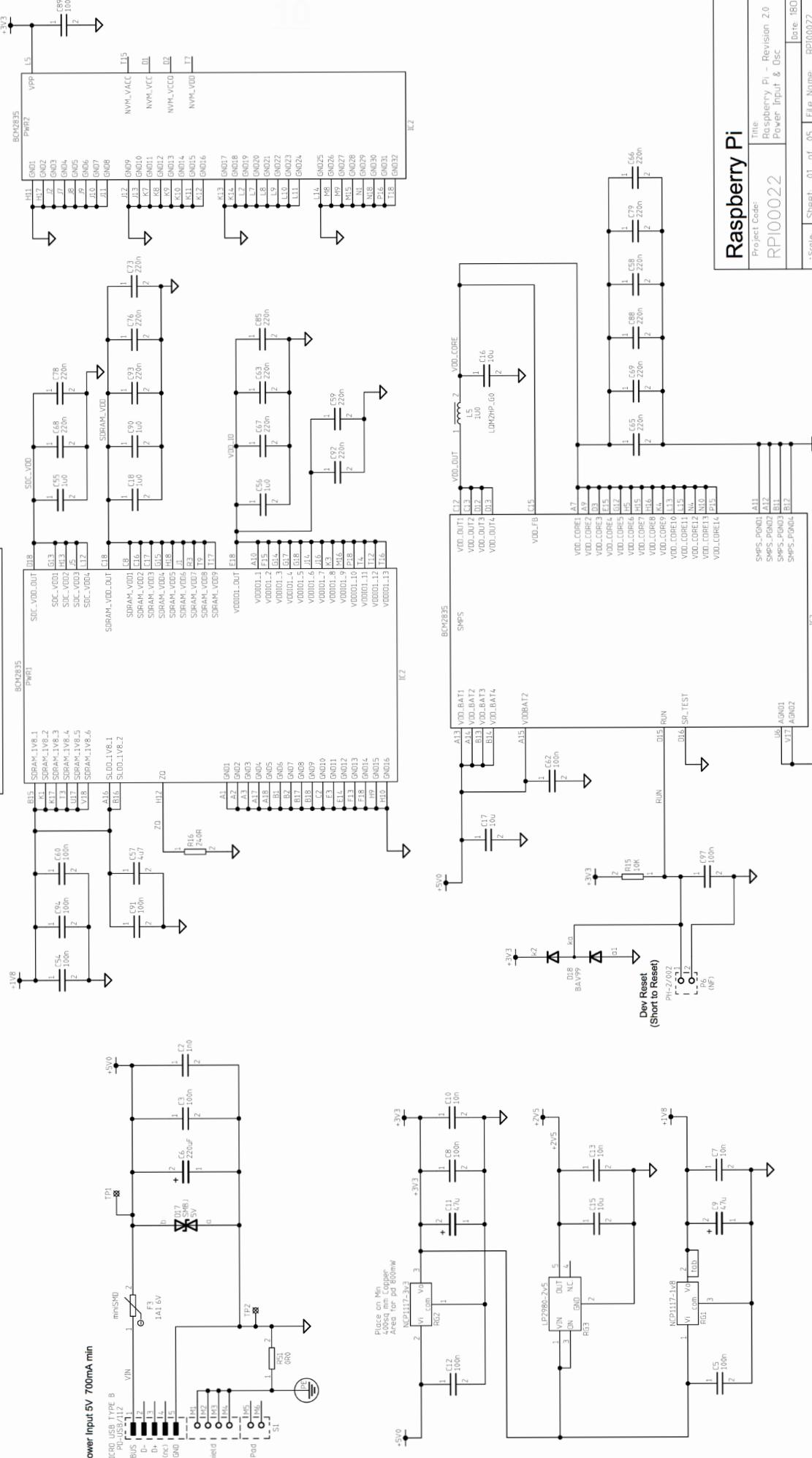
#mon_fichier = open("etatrelais.txt", "r")
#etatrelais = mon_fichier.readlines()
#print etatrelais

#etat = str(bus.read_byte(address))
#print "relai 1:"
#print etat[0]
#print "relai 2:"
#print etat[1]
#print "relai 3:"
#print etat[2]

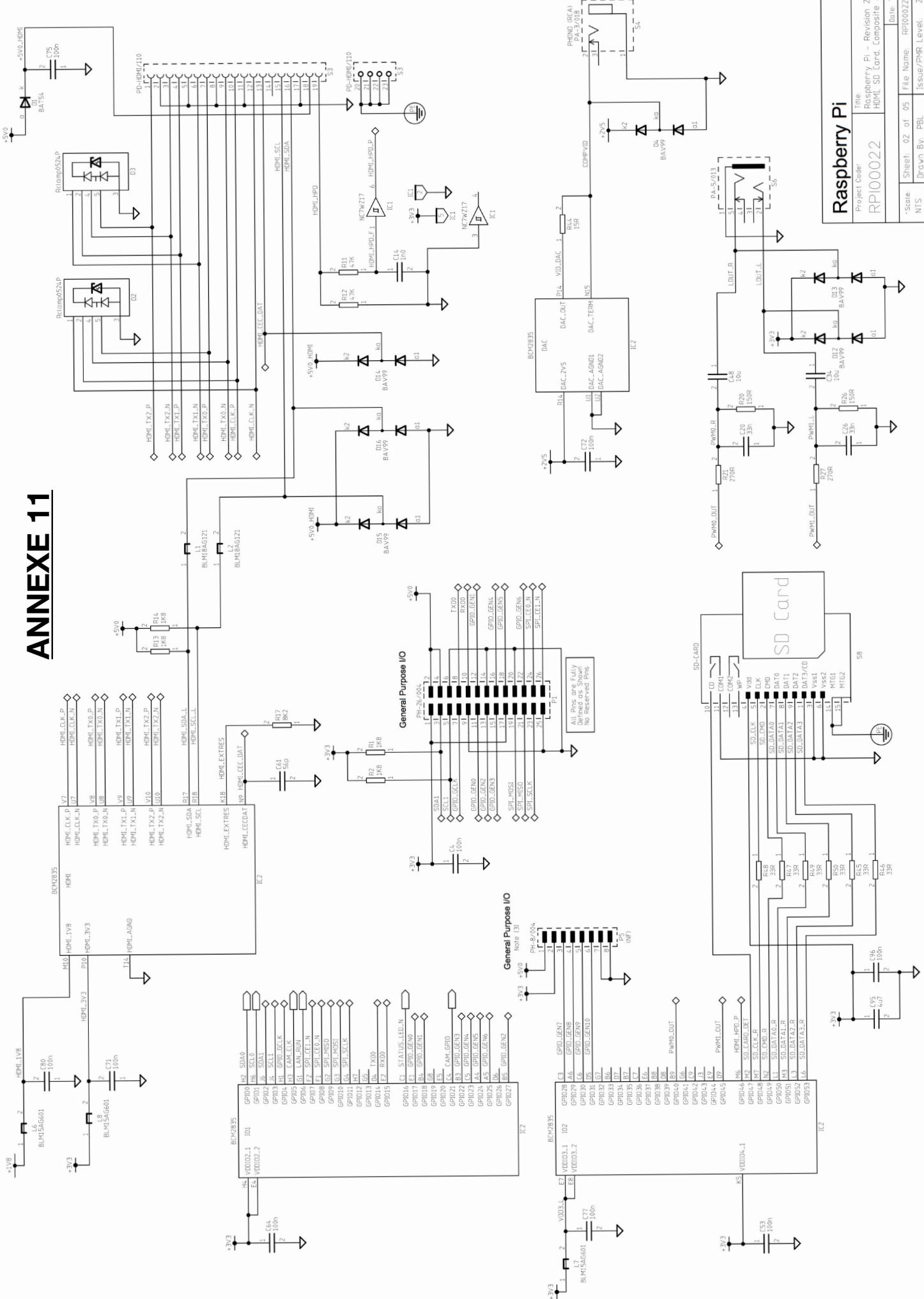
#-----#
#print "<p></p>" 
#print "-----<p></p>" 
#print "-", "Relai 1", "|", etat[0], "|<p></p>" 
#print "----+----+----<p></p>" 
#print "-", "Relai 2", "|", etat[1], "|<p></p>" 
#print "----+----+----<p></p>" 
#print "-", "Relai 3", "|", etat[2], "|<p></p>" 
#print << -----<p></p>"
```

ANNEXE 10

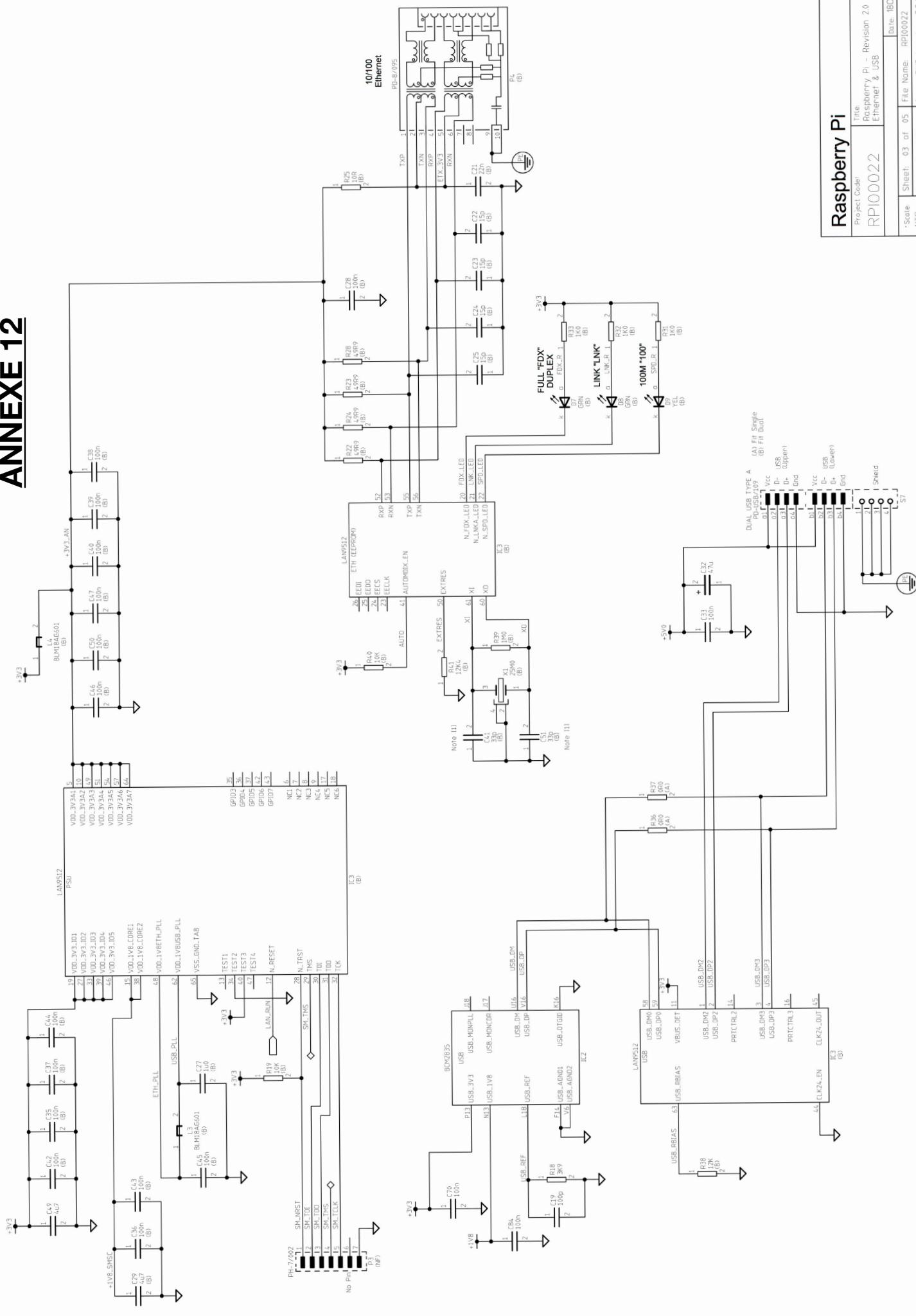
MEMORY: IC2A fitted as PDP to IC2 - Note [2]
 256Byte = K4P 2 G324EF
 512Byte = K4P 4 G324EF



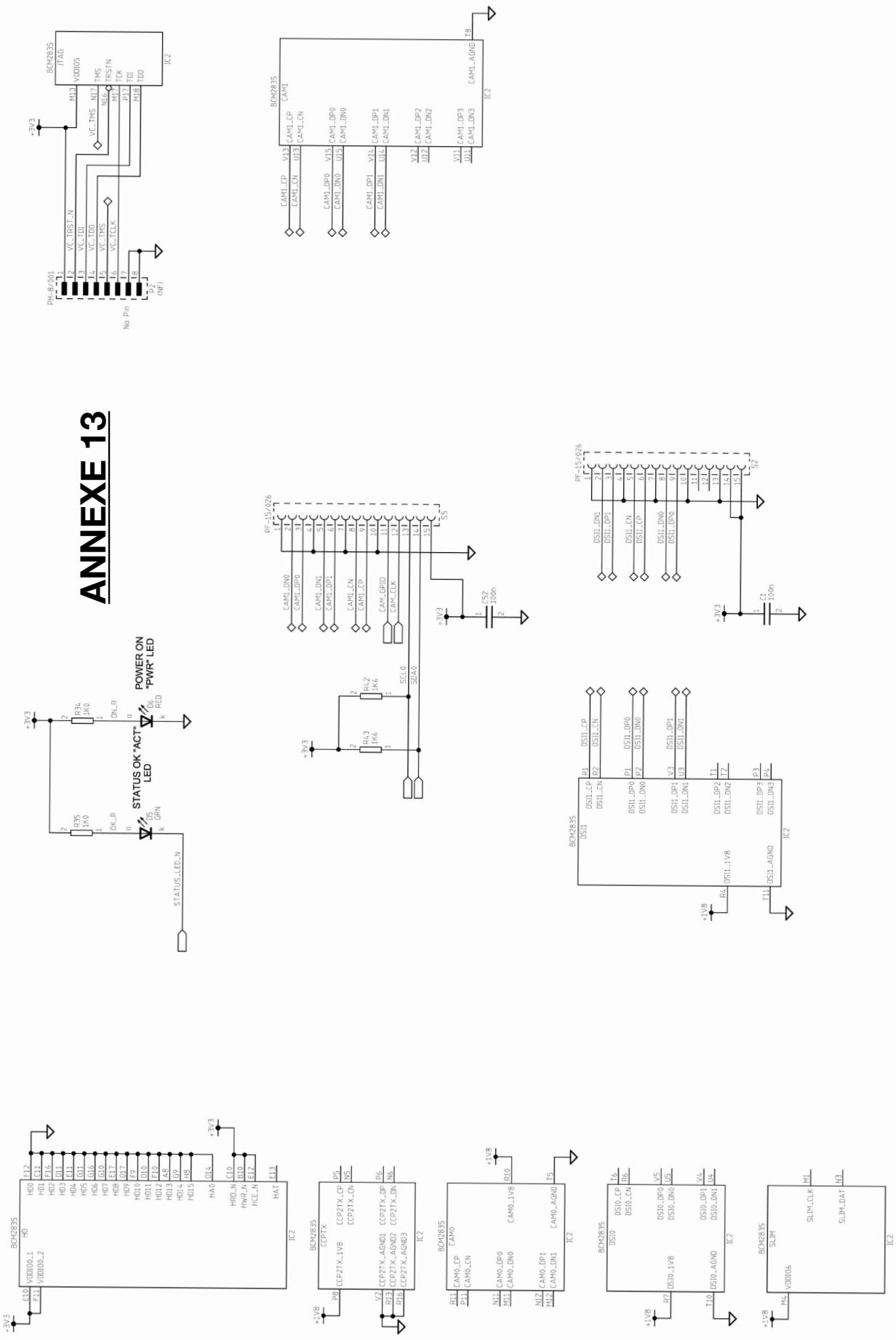
ANNEXE 11



ANNEXE 12



ANNEXE 13



Raspberry Pi

Project Code: RP00022 Date: 18/04/2022

Title: Raspberry Pi - Revision 2.0
User: NTS Date: 18/04/2022

Issue/PMR Level: 2.2/(27)

ANNEXE 14

PCB Layout Requirements	
Layout notes referring to specific components or groups of components are indicated on the schematic by the note number enclosed in brackets adjacent to the component. Eg 1(a).	
Note No.	Description
1	These are general notes relating to the build of the product.

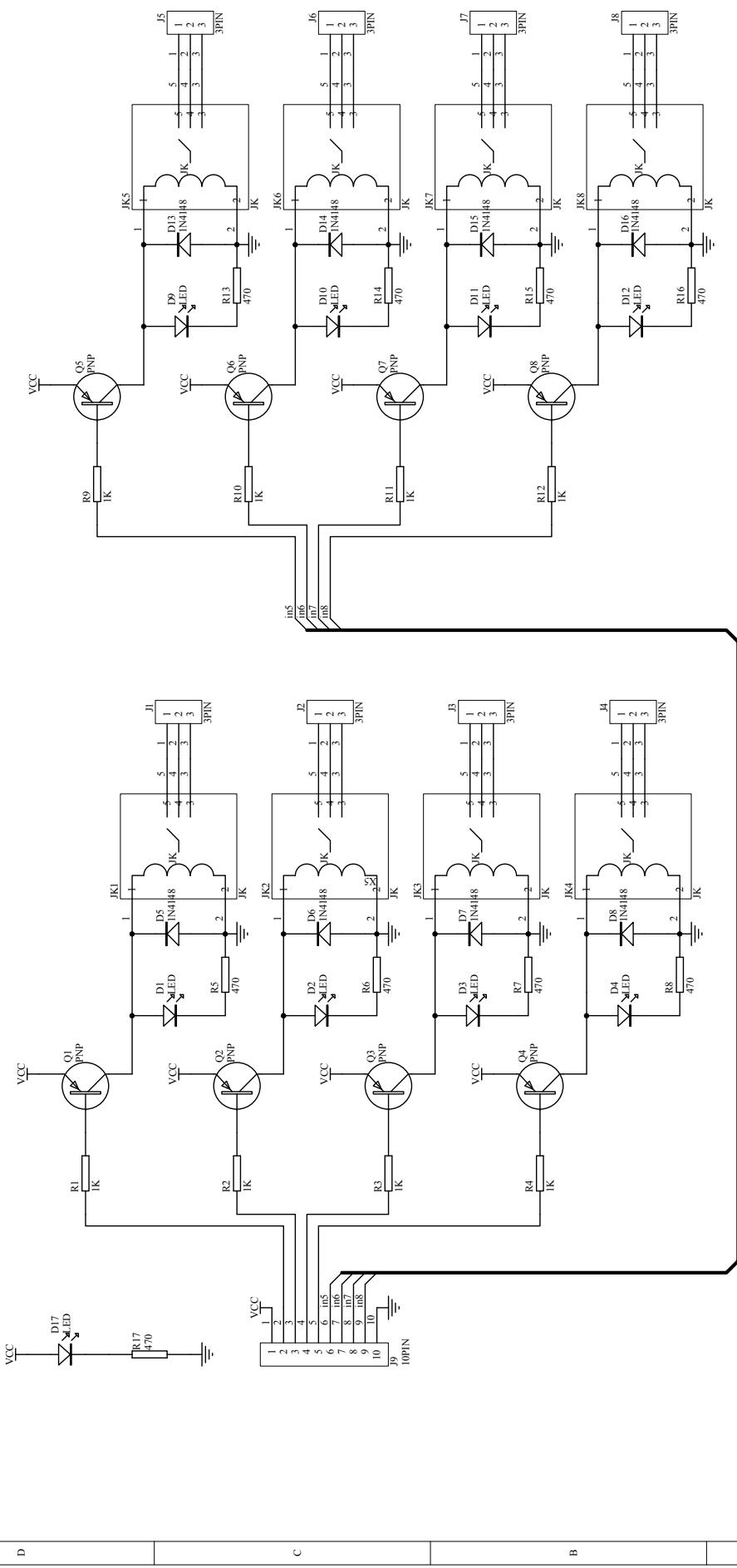
PCB Build Variations	
Each Build variant is assigned a letter A-Z, letter, in parentheses, associated with it on the schematic pages.	
Build Variant	Build Description
2	

PCB Assembly Options	
Notes: A PCB assembly option ('Build Option') is formed by summing the required variants of circuit sections. Eg A+B+C	
Build variants	Build Description
A	Components only fitted to Model A (Basic)
B	Components only fitted to Model B (Full System)

PCB Parts List Entry	
= Project Date = Issue	
Part Ref	Description
BD-RPI-00022/001	

Raspberry Pi	
Project Code:	Title:
RPI00022	Raspberry Pi - Revision 2.0
	Build Options/PCB layout Instructions
	Date: 18/03/12
Client n/a	Sheet: 05 of 05
Scale:	File Name:
N/A	RPI0022
	Issue/FMR Level: 2/2(027)

ANNEXE 15



Title		YwRobot	
Size	Number	Revision	
B			
Date:	8-Jun-2011	Sheet of	1084
File:	F:\U\H\G\T\Y\W\YwRobot\Design.addb	1084	YwRobot

1 2 3 4

1 2 3

1 2 3

1 2 3

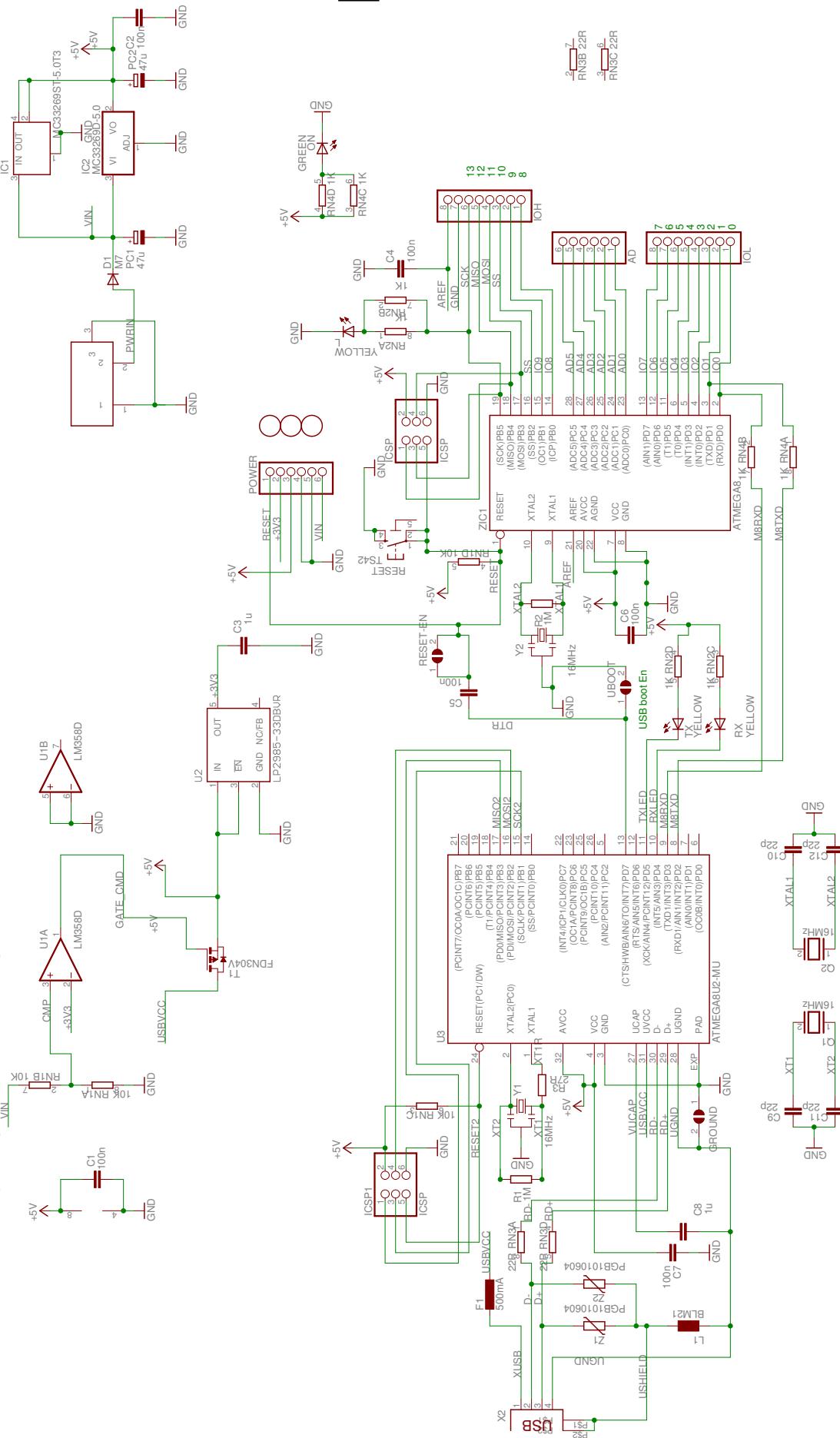
Arduno™ UNO Reference Design

Reference Designs ARE PROVIDED "AS IS" AND "WITH ALL FAULTS". Arduino DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Arduno may make changes to specifications or product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Arduino reserves these terms for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Material is subject to change without notice. Do not include a design with this information.

ANNEXE 16

16



ANNEXE 17

Nom	Date de début	Date de fin
▼ • Conception préliminaire		
• étude des solution possible pour la gestion à distance	01/11/13	13/12/13
• choix des solutions	01/11/13	01/11/13
• Bilan des solutions envisagées	04/11/13	04/11/13
• répartition du travail		
• élaboration des diagrammes fonctionnels SysML	08/11/13	08/11/13
▼ • Conception détaillée		
• Choix des sous ensemble constituants et composant de la cart...	15/11/13	17/02/14
• choix des sous ensemble constituant et composant de l'interfac...	15/11/13	15/11/13
• définir une structure logicielle pour l'interface web	15/11/13	15/11/13
• regroupement des parties individuelles	15/11/13	29/11/13
• simulation des fonctions répondant aux solution choisies	03/02/14	03/02/14
▼ • Maquettage-prototypage		
• Commande et réception du matériel	15/11/13	06/12/13
• Réalisation et contrôle la commande - Implantation du progra...	02/12/13	14/02/14
• Site Web	15/11/13	07/03/14
• Configuration du Raspberry Pi	02/12/13	24/02/14
• Création d'une application Smartphone (IOS)	03/03/14	01/04/14
• Création d'une application Smartphone (Android)	19/03/14	11/04/14
• Vérification du fonctionnement des applications	31/03/14	11/04/14
• Vérification du fonctionnement et de la conformité de chaque p...	31/03/14	11/04/14
• Assemblage des parties individuelles	31/03/14	11/04/14
• Mise en oeuvre du projet et test.	31/03/14	11/04/14
▼ • Test et validation		
• Mise en place d'un protocole d'essai	10/03/14	28/03/14
• Essais et mesures des performances	10/03/14	28/03/14
• Analyse des résultats et bilan technique	17/03/14	04/04/14
• Proposition d'amélioration	24/03/14	11/04/14

