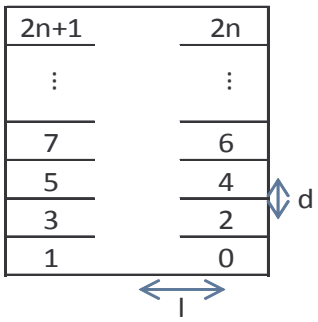
Conception des programmes

Les programmes se diviseraient en trois parties et en trois langages de programmation différents. Nous avons défini un embryon d'architecture, mais devons nous renseigner sur ces différents langages afin de voir si nous pourrons réaliser les choses de la façon dont nous les voyons.

Vous trouverez dans les pages à venir les programmes tels que nous les imaginons s'articuler entre eux. La première partie concernera les capteurs. Le programme sera en langage nesC. La seconde partie, qui devra être réalisée en C, portera sur le programme du WiFiBot. Enfin, nous pensons que la troisième partie devra être un fichier exécutable dans la console Cygwin, donc rédigée en langage shell, mais une fois encore, nous devons pousser plus loin les recherches sur les possibilités de ce langage avant d'affirmer pouvoir le réaliser de la façon qui va être présentée ci-dessous.

Ce qui est présenté ci-dessous est optimisé pour la disposition suivante :



Nous tenons à rappeler que ces pages ne sont pas des programmes mais seulement des « schémas » d'architecture. Leur forme est cependant proche d'un programme car c'est la façon la plus simple de schématiser. Ces schémas devraient être compréhensibles par toute personne ayant des notions de base de programmation en java et en C.

Toutes les parties précédées d'un ** désignent des fonctions qu'il était plus simple d'expliquer sous forme de phrase que sous forme d'un pseudo-langage de programmation.

Capteur

```
# DEFINE REF //signal de référence

main() {
      ** le capteur attend un signal
      if (** reçoit un signal s)
            if (s == REF)
                  ** le capteur envoie l'état de la lumière
}
```

WiFiBot

```
# DEFINE L //distance d'un emplacement à l'allée centrale
# DEFINE D //écart entre deux emplacements

main() {
      ** le robot attend les ordres et les exécute à l'aide des
      fonctions ci-dessous
}

procéder(int pos, int r) {
/* Cette fonction permet d'aller déposer ou rechercher quelque chose à
l'emplacement pos */
      avancer(L);
      if (pos == 1) {
            tour(2);//le robot se met dos à l'emplacement
            if (r == 1)
                  dépôt();
            else
                  retrait();
            avancer(L);//le robot revient sur son emplacement
            tour(2);//le robot est prêt à repartir
      } else {
            //on détermine les variables de la procédure
            int d;
            if (pos%2 == 1)
                  d = 3;
            else
                  d = 1;
            avancer(L);//le robot va au milieu de l'allée
            tour(3);//le robot tourne à droite
            avancer(D*(pos/2));//le robot va au niveau de l'emplacement
            tour(d);// le robot tourne le dos à l'emplacement
            if (r == 1)
                  dépôt();
            else
                  retrait();
            tour(d);//le robot se prépare à rentrer
            avancer(D*(pos/2));
            tour(1);
            reculer(L);//le robot est dans sa position initiale
      }
}


avancer(float d) {
      ** le robot avance d'une distance d
}

reculer(float d) {
      ** le robot recule d'une distance d
}
```

```
tour(int d) {
      if (d == 1)
            ** quart de tour vers la gauche
      else if (d == 2)
            ** demi-tour
      else if (d == 3)
            ** quart de tour vers la droite
      else
            error();
}

error() {
      int n = 0;
      float d = min(L,D)/5;//le robot ne doit pas heurter les rayonnages
      while (n < 5) {
            avancer(d);
            reculer(d);
            n++;
      }
}

dépôt() {
/* permet au robot de déposer une palette s'il tourne le dos à l'emplacement */
      reculer(L);
      ** le robot dépose son objet
      avancer(L);
}

retrait() {
/* permet au robot de retirer une palette s'il tourne le dos à l'emplacement */
      reculer(L);
      ** le robot retire son objet
      avancer(L);
}
```

Station de travail

```
# DEFINE REF //signal de référence
# DEFINE CAP //nombre de capteurs

main() {
      ** la station attend une commande et redirige vers la bonne
      fonction, qui peut être un autre exécutable dans le cas de
      l'utilisation de langage shell
}

dépôt() {
      ** la station demande d'entrer le code du produit qui va être
      stocké
      ** envoi du signal REF
      ** récupération de l'état des capteurs 1 à 7
      int s = 1 ;
      boolean f = false;
      while (s < CAP + 1) {
            if (** la station s est libre) {
                  f = true;
                  break;
            }
      }
      if (f) {
            ** envoi au robot de l'ordre procédure(s,1)
            ** stockage du code produit dans la base de donnée à
            l'emplacement s
      } else {
            ** envoi au robot de l'ordre error()
            ** affichage d'alerte : "plus de place dans l'entrepôt"
      }
}

retrait() {
      int s = 1 ;
      while (s < CAP + 1) {
            ** la station affiche : s + ". " + codeproduit
      }
      ** l'utilisateur doit sélectionner le produit qui l'intéresse
      ou annuler
      if (** l'utilisateur choisit un produit)
            ** envoi au robot de l'ordre procédure(s,2)
}
```