|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **SUPPORT** | | | **CLASSE :** |
| **Radar Pédagogique**  **Icare-M** | | | 🗹 Première  🗹 Terminale |
| **ACTIVITE** | **Enseignement Technologique Transversal** | **Enseignement Technologique de Spécialité** | **Thème**  **Sociétal** | |
| 🞏 E.D.T.  🗹 Activité pratique  🞏 Projet | 🞏 Matière  🞏 Energie  🞏 Information | 🗹 S.I.N.  🞏 E.E.  🞏 I.T.E.C.  🞏 A.C. | **Améliorer le confort d’un environnement** | |
| **Durée** | **Centre(s) d’Intérêt(s) :** | | | |
| 4H | **CI3** Communication de l’information | | | |
| **Objectif(s) :** | O7 - Imaginer une solution, répondre à un besoin  O8 - Valider des solutions techniques | | | |
| **Compétences attendues :** | CO7.sin1. Décoder la notice technique d’un système, vérifier la conformité du fonctionnement  CO7.sin3. Exprimer le principe de fonctionnement d’un système à partir des diagrammes SysML pertinents.  CO8.sin1. Rechercher et choisir une solution logicielle ou  matérielle au regard de la définition d'un système | | | |
| **Pré-requis :** | - Numération  - Algorithme  - Langage de programmation arduino | | | |

**Données et conditions** :

* Carte texte, afficheur à matrice de Leds et carte arduino.
* Rendre le TP à la fin de la séance.

**Mise en situation :**

Le système Radar Pédagogique Icare-M mesure la vitesse des véhicules et affiche des informations à l’aide d’un afficheur à matrice de Led.

Ce TP porte sur la fonction "afficher", correspondant au diagramme (ou analyse par SysML) spécifié dans le chapitre "approche fonctionnelle" du dossier technique. Il concerne l’analyse de la gestion de l’affichage par le microcontrôleur.

**Problématique** :

**Comment afficher des symboles ou des lettres ?**

# Approche fonctionnelle du système « radar pédagogique »

**Q1 :** Le radar présent dans la salle est le radar ICARE-M. Le fichier pdf « plaquette\_icare.pdf » présente le radar I-CARE . Quelle est la différence entre les deux radars en ce qui concerne l’affichage ?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Q2 :** Compléter le diagramme des exigences du radar ICARE-M en annexe pour prendre en compte la différence du radar I-CARE.

# Affichage des caractères sur la matice de leds

Dans la suite de cette activité, vous comprendrez tout d’abord comment est gérée la matrice de leds supplémentaire du radar I-CARE. Ensuite, vous afficherez un caractère « P » puis un caractère ou un symbole de votre choix et pour finir un mot de 4 lettres.

## Présentation

Les composants qui gèrent la matrice de leds sont constitués de registres à décalage mis en série, commandés par un signal d’horloge. Il y a donc une entrée série (SDI) et des sorties parrallèles.

La carte est constituée de 24 x 8 sorties parrallèles. Sur chaque groupe de 8 sorties, les 7 premières sorties sont relièes à une led, la 8ème sortie n’est pas reliée. Chaque groupe de 7 leds constitue une colonne sur la carte d’affichage. Il y a donc 24 colonnes au total.



1ère led

Dernière led

24ème colonne

1ère colonne

Trois sorties sont donc nécessaires pour pouvoir commander l’affichage à matrice de Leds.

* SDI : Permet de décider si une led va être allumée (SDI=1) ou etteinte (SDI=0)
* CLK : Horloge de commande du registre à décalage.
* LE : Latch enable, commande permettant de recopier le contenu du registre à décalage sur les sorties parrallèles du composant pour mettre à jour l’état des leds.

A chaque front montant de l’horloge CLK, l’état d’une led est transféré à la led suivante. L’état des 168 leds est mis à jour au même moment. Pourvisualiser le nouvel état de chaque led, il faut alors générer une impulsion sur le signal LE.

Exemple :

Etat initiale : toutes les led sont etteintes

SDI

CLK

LE

Résultat :



Ensuite en appliquant les signaux suivants :

SDI

CLK

LE

Résultat :



Les signaux suivants sont ensuite appliqués :

SDI

CLK

LE

Q3 : Déterminer l’état des leds ci-dessous :



On continue avec les signaux suivants :

SDI

CLK

LE

Q4 : Déterminer l’état des leds ci-dessous (Rappel : La huitième sorties n’est pas reliée à une led) :



## Affichage du caractère « P ».

Cette matrice est prévue pour afficher 4 caractères, chaque caractère occupera donc une place de 6x7 leds. Exemple pour le caractère P (Rappel : 1 pour allumer, 0 pour éteindre) :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |

Vous allez maintenant essayer d’afficher le caractère P en commençant par afficher une première colonne.

void loop() {

digitalWrite(SDI,LOW);

digitalWrite(CLK,HIGH);

digitalWrite(CLK,LOW);

digitalWrite(SDI,LOW);

digitalWrite(CLK,HIGH);

digitalWrite(CLK,LOW);

digitalWrite(SDI,HIGH);

digitalWrite(CLK,HIGH);

digitalWrite(CLK,LOW);

digitalWrite(SDI,HIGH);

digitalWrite(CLK,HIGH);

digitalWrite(CLK,LOW);

digitalWrite(SDI,LOW);

digitalWrite(CLK,HIGH);

digitalWrite(CLK,LOW);

digitalWrite(SDI,LOW);

digitalWrite(CLK,HIGH);

digitalWrite(CLK,LOW);

digitalWrite(SDI,LOW);

digitalWrite(CLK,HIGH);

digitalWrite(CLK,LOW);

digitalWrite(SDI,LOW);

digitalWrite(CLK,HIGH);

digitalWrite(CLK,LOW);

digitalWrite(LE,HIGH);

digitalWrite(LE,LOW);

while(1);

}

Q5 : Par quelle colonne devez-vous commencer ?

La première colonne à droite\_\_\_\_\_\_\_\_\_

Q6 : En analysant le code de la fonction loop ci-contre, complétez les chronogrammes page suivante.

Q7 : Compléter la fonction loop du sketch « affiche0.ino » avec le code ci-contre dans votre environnement « arduino ».Compiler et téléverser le programme dans le module arduino.

Q8 :Le résultat est-il conforme à votre attente ? Pourquoi utilise-t-on huit coup d’horloge ?

Réponses Q6 :

SDI

CLK

LE

Q9 : Compléter le programme de manière à afficher complètement le caractère P. Appeler le professeur lorsque vous aurez réussi. (Voir affiche2.ino)

## Première optimisation du programme

Il est possible de prédéfinir le caractère dans un tableau :

const byte P[48]={

0,0,1,1,0,0,0,0,

0,1,0,0,1,0,0,0,

0,1,0,0,1,0,0,0,

0,1,0,0,1,0,0,0,

0,1,1,1,1,1,1,1,

0,0,0,0,0,0,0,0} ;

Le programme principal devient alors :

void loop() {

char i;

for(i=0;i<48;i++){

if (P[i]==1)

digitalWrite(SDI,HIGH);

else

digitalWrite(SDI,LOW);

digitalWrite(CLK,HIGH);

digitalWrite(CLK,LOW);

}

digitalWrite(LE,HIGH);

digitalWrite(LE,LOW);

while(1);

}

Q10 : Quelles valeurs prendra la variable i ?

Les valeurs 0 à 47.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Q11 : Pour quels indices (valeurs de i) P[i] sera-t-il égale à 1 ?

2, 3, 9,12, 17, 20, 25, 28, 33, 34, 35, 36, 37 et 39.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Q12 : Charger le programme « affiche3.ino » et exécuter le. Le caractère s’affiche-t-il correctement ?

Oui\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Q13 : Configurer le caractère de votre choix dans le tableau ci-dessous (Ne pas oublier que la première colonne est utilisée pour séparer deux caractères juxtaposés) :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |

Q14 : Mettre à jour le tableau dans le programme précédent et valider votre caractère. Appeler le professeur lorsque votre caractère s’affiche correctement.

## Deuxième optimisation du programme

Il est inutile d’utiliser autant de place mémoire, on peut donc optimiser le tableau en utilisant les 8 bits de chaque octet du tableau. Chacun de ces octets correspondra donc à une colonne de votre caractère. Pour le caractère P, le nouveau tableau est le suivant :

const byte P[6]={ 0x30,0X48,0x48,0x48,0x7F,0x00} ;

Le programme principal devient alors :

void loop() {

char Test,i,j;

for(i=0;i<6;i++){

for(j=0;j<8;j++) {

Test=(P[i]>>(7-j))&0x01;

if (Test==0x01)

digitalWrite(SDI,HIGH);

else

digitalWrite(SDI,LOW);

digitalWrite(CLK,HIGH);

digitalWrite(CLK,LOW);

}

}

digitalWrite(LE,HIGH);

digitalWrite(LE,LOW);

while(1);

}

La syntaxe (P[i]>>n) permet de faire un décalage à droite **n** fois du contenu de la variable P[**i**]. Pour chaque décalage, un 0 est placé sur le bit de poids fort.

Q15 : Remplir les valeurs ci-dessous lorsque i=1 et j= 0 à 4 :

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P[i] | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P[i]>>(7-0) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P[i]>>(7-1) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P[i]>>(7-2) | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P[i]>>(7-3) | 0 | 0 | 0 | 0. | 0. | 1 | 0 | 0 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P[i]>>(7-4) | 0 | 0 | 0 | 0. | 1 | 0 | 0 | 1 |

Un « et » avec l’octet 0x01 est ensuite appliqué avec le résultat du décalage précédent.

Q16 : Complétez le tableau ci-dessous :

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P[i]>>(7-4) | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0x01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| (P[i]>>3)&0x01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Q17 : Quel sera le résultat final si au départ le bit de poid (7-j) de P[i] était égal à 1 ?

0x01\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Q18 : Quel sera le résultat final si au départ le bit de poid (7-j) de P[i] était égal à 0 ?

0x00\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Q19 : Expliquer alors quel est l’utilité du « if (Test==1) »

Détecter la présence d’un « 1 » pour choisir l’état de la led correspondante\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Q20 : Faire l’organigramme de la fonction « loop ».

I=0

Non

Non

Non

Oui

Oui

Oui

i=i+1

Fin

Latch

j=0

I<6

j=j+1

Un coup d’horloge

SDI=0

SDI=1

Test=1

Affecter à Test la valeur du (7-j)eme bit du ième octet du tableau P

j<8

Q21 : Chargez le programme « affiche4.ino », exécutez le. Le caractère s’affiche-t-il correctement ?

Oui\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Q22 : Mettez à jour le tableau de façon à faire apparaitre votre caractère de la question 13. Tester votre tableau en modifiant le programme « affiche4.ino »

const byte P[6]={0x38,0x44,0x22,0x44,0x38,0x00} ;

# Synthèse : comment afficher le mot « STOP » ?

Q23 : Combien de coup d’horloge sont nécessaires pour afficher directement 4 caractères ?

192 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Q24 : Proposez un tableau et modifiez la fonction « loop » de façon à afficher le mot STOP sur l’afficheur : (Voir fichier affiche5.ino)

const byte P[24]={ 0x30,0X48,0x48,0x48,0x7F,0x00,0x3E,0x41,0x41,0x41,0x3E,0x00,0x40,0x40,0x7F,0x40,0x40,0x00,0x26,0x49,0x49,0x49,0x32,0x00}

Diagramme des exigences :

