Table des matières

L'arduino

Qu'est ce que l'arduino

L'arduino est un système de prototypage électronique open source basé sur plusieurs micro contrôleur. Ce sont des cartes électronique, à l'origine développées par Arduino (le projet étant open source, l'on trouve aujourd'hui pas mal de clone compatible fabriqué par des concurrent), qui se programme directement dans un langage proche du C.

Le sketch (le code) ainsi programmer se charge directement en usb dans la carte qui se comporte après comme une carte autonome ou contrôlable avec divers protocoles. Suivant la carte un certain nombre d'entrée analogique ou numérique et de sortie PWM (gradué) ou numérique (0 ou 1) est configurable.

Pour de plus ample informations sur le système, je vous renvoie vers: https://www.arduino.cc/ site du constructeur.

https://www.arduino.cc/en/Guide/HomePage guide en anglais.

http://playground.arduino.cc/ Le playground: énorme référence d'interfaçage électronique entre l'arduino et divers composant. Presque pour chaque cas d'utilisation vous trouverez votre bonheur.

https://www.arduino.cc/en/Reference/HomePage référence du langage.

http://eskimon.fr/58-arduino-1-decouverte-de-larduino un chouette tutoriel pour prendre l'arduino de zéro.

Et naturellement l'excellent wiki de WhiteCat avec qui j'ai fait mes premiers pas avec l'arduino. http://www.le-chat-noir-numerique.fr/whitecat/dokuwiki/doku.php?id=configuration_arduino&DokuWiki=ee016dcf9e0c5cba16ea4be6b3c1b7e8

Les entrés sorties de l'arduino

Chaque carte Arduino à un nombre définit d'entrées/sorties. Ce sont les pins sur lesquelles vous pourrez brancher votre circuit. La duemilanove ou la Uno ont part exemple 13 entrées/sorties numériques et 6 entrées analogiques.

Les entrées

Dans Héphaistos, les entrées peuvent être de trois type.

1. Analogique: permet de lire un potentiomètre ou un capteur. Elle renvoie une valeur entre 0 et 1023.
2. Numérique: permet de lire l'état d'un bouton ou tout circuit ouvert/fermé. Elle renvoie soit 0, soit 1.
3. Pull-up: En général pour brancher une entrée numérique l'on a besoin d'un circuit dit pull-up ou pull-down. C'est en fait une résistance branche au +5v de l'alimentation ou à la masse pour éviter que le contact avec l'entrée ne provoque un court circuit franc, une intensité illimité et de

brule la carte.

Heureusement pour nous, la plupart des microprocesseur comme l'ATMEGA328 sur lequel est basé la UNO possède un pull-up intégrer. Mettre une entrée en mode pull-up nous permet de faire le montage suivant, qui est nétement plus pratique et évite une résistance, pour brancher par exemple un bouton poussoir.

Les sorties

Les sorties peuvent avoir 2 mode:

1. Les sorties numériques : met la valeur de la sortie à 0v ou +5v
2. Les sorties PWM (pulse width modulation): c'est les sortie gradué entre 0 et 255. Par défaut sur un microcontroleur les sortie ne gradue pas entre 0 et 5V, il s'agit en fait de PWM.

C'est à dire que la sortie va passer de l'état 5V à 0V de manière continue et pendant plus ou moins longtemps (très mal expliqué mais aller voir https://fr.wikipedia.org/wiki/Modulation_de_largeur_d%27impulsion si vous voulez plus de détail). C'est finalement le principe de tout gradateur.

Attention néanmoins par défaut sur les cartes arduino, seul certaines pattes ont la capacité de générer un signal PWM. La librairie softPWM https://code.google.com/p/rogue-code/wiki/SoftPWMLibraryDocumentation permet d'émuler le signal sur toute les sorties de l'arduino.

Les librairies

Comme dans la plupart des langages informatiques l'on peut charger des librairies dans le code arduino. Une librairie est en fait tout une partie de code, créer par un autre développer, que l'on peut inclure directement en une ligne pour éviter d'avoir à tout réécrire. Une fois la librairie inclue, l'on pourras directement appeler les fonctions de celle-ci, qui appellerons les bouts de code créer par un autre développer.

L'on trouve des librairies pour la plupart des fonctions, allant de la communication en dmx, controle des moteurs, hack de l'arduino pour émuler des sorties pwm sur tout les pins de l'arduino etc.. Vous en trouverez beaucoup dans le lien du playground ci -dessus.

installer la librairie HephaistosArduinoLib

Après plusieurs approche dans le suivit du développement de whitecat (du sketch par défault à réécrire, allant d'un module sans ligne de code permettant de configurer et contrôler l'arduino directement depuis whitecat), pour permettre à l'utilisateur final de prototyper des appareil qui communique avec le logiciel, j'ai finalement choisis l'approche de développer directement une librairie afin de gérer la reconnaissance et la communication en port serie (usb) entre Héphaïstos et l'arduino. Vous la trouverez dans le dossier /ressources du repertoire d'Héphaïstos.

Pour installer la libraire dezipper simplement le dossier dans le répertoire /libraries de votre IDE (programme) arduino ou directement depuis l'IDE dans l'onglet croquis/importer bibliothèque/ajouter bibliothèque.

Au prochain démarrage du logiciel arduino vous devrez trouver dans l'onglet fichier/Exemples la librairie HephaistosArduinoLib et vous pourrez ouvrir l'exemple pwmsimple.

Les principales fonction de la librairie HephaistosArduinoLib

- HEPHAISTOS heph = HEPHAISTOS(); permet de créer un objet de communication usb entre Hephaistos et l'arduino

Chaque fonction de la librairie font être appelé avec cet objet heph (ex:heph.begin(Serial, BAUDRATE);)

- void begin(Stream &serial, const int baudrate) initialise la libraire et définit le port serie ainsi que le baudrate de comunication
- void setSerial(Stream &serial, const int baudrate) permet de redéfinir un autre port série.
- void checkSerial() doit être appeler régulièrement. Vérifie la communication série et traite les données.
- byte HEPHAISTOS::getFromHephaistos(int num) permet de retourner la valeur de la Xième sortie (num) envoyées par Héphaïstos. (ex: getFromHephaistos(3) renvoie la valuer entre 0 et 255 de la troisième sortie).
- void setToHephaistos(byte buffer[], int buffersize) permet d'envoyer un tableau de valeur vers les entrées d'Héphaïstos.

Le code de pwmsimple

Le code d'exemple permet de configurer les pins 2 à 13 de l'arduino en sortie PWM et les pins A0 à A2 en entré analogique et A3 à A5 en entrés numérique avec pull-up.

La partie du texte grisé précéder de // sont des commentaires. Il ne font pas parties du code mais permettent simple de l'expliquer un peu à la lecture.

```
#include <Hephaistos.h>

#include <SoftPWM.h>
#include <SoftPWM_timer.h>


HEPHAISTOS heph = HEPHAISTOS(); //créer un objet HEPHAISTOS qui servira à la comunication en série avec le programme

////////////////A adapter par l'utilisateur//////////////////////////////////
const int BAUDRATE=9600; //vitesse de transmition sur le port série. Doit être identique dans la fenêtre de configuration arduino de Héphaistos.
//definition des input
const int NBR_IN=6;  //Nombre de patte en entrées. Doit être identique dans
```

```
la fenêtre de configuration arduino de Héphaistos.
int input[NBR_IN]={A0,A1,A2,A3,A4,A5}; //definit les numéros des pins
utilisé en entré
boolean is_pullup[NBR_IN]={false,false,false,true,true,true}; //pour chaque
pins définies plus haut: true pour pouvoir relier la pin à la masse et se
passer de résistances
boolean is_analog_in[NBR_IN]={true,true,true,false,false,false}; //pour
chaque pins définies plus haut: true s'il s'agit d'entrés analogiques

///Definition des sortie
const int NBR_OUT=12; //Nombre de patte en sorties
int output[NBR_OUT]={2,3,4,5,6,7,8,9,10,11,12,13};  //definit les numéros de
pin utilisé en sortie
boolean
is_PWM[NBR_OUT]={true,true,true,true,true,true,true,true,true,true,true,true
}; //true si la sortie est PWM

//////////////////SUITE DU
CODE//////////////////////////////////////////////////////


//////////////////////Definition des parametre d'input
int threshold_analog=0; //la difference necessaire pour que les changement
soit pris en compte. L'augmenter permet de ne pas prendre en compte les
micro vraiation de valeur.
int sensorMin = 1023;        // minimum sensor value
int sensorMax = 0;           // maximum sensor value

byte buffer_digital[NBR_IN]; //tableau ou sont stocker les valeurs des
entrés digitales
byte old_buffer_digital[NBR_IN]; //anciennes valeurs avant lecture. permet
de détecter s'il y a eu un changement pour éviter d'envoyer les valeurs en
continu.
boolean buttonState[NBR_IN]; // contient les etats (ouvert/fermé) des entrés
numériques

void setup()
{
     Serial.begin(BAUDRATE); //initialise la comunication série au baudrate
définis
     SoftPWMBegin();//initialise le PWM sur toutes les sorties
     heph.begin(Serial, BAUDRATE); //initialise l'objet de communication
avec Hephaistos en lui indiquant le port série utilisé et le baudrate

 for(int i=0; i<NBR_OUT; i++)
{
 pinMode(output[i], OUTPUT);  //définit toutes les pins contenu dans le
tableau output en sortie
}
//initialisation des entrées
for(int i=0; i<NBR_IN; i++)
```

```
{
  pinMode(input[i], INPUT); //définit toutes les pins contenu dans le
tableau input en entré

 //définit le mode pull up pour chaque entrée.
  if(is_pullup[i]&&is_analog_in[i]==false)
  {digitalWrite(input[i], HIGH);}
  else
  {digitalWrite(input[i], LOW);}

}

}

void loop()
{
heph.checkSerial(); //vérifie les instructions sur le port série. important
en début de boucle
read_input(); //fonction définie plus bas de lecture des entrées

//pour chaque sorties écrit la valeur envoyée par Hephaistos en pwm
for(int i=0;i<NBR_OUT; i++)
 {
   if(i<heph.hephBufferSize()) //nombre de sorties envoyer par Hephaistos
 {
 SoftPWMSet(output[i],heph.getFromHephaistos(i)); //ecrit sur la sortie
output[i] la valeur envoyer par Hephaistos heph.getFromHephaistos(i)
 }
 }
}

void read_input()
{
  for(int i=0; i<NBR_IN; i++)
  {
        old_buffer_digital[i]=buffer_digital[i];

    if(is_analog_in[i]==false)
   {
    if(is_pullup[i])
    {buttonState[i]= !digitalRead(input[i]);}
    if(is_pullup[i]==false)
    {buttonState[i]=digitalRead(input[i]);}

      if(buttonState[i])
    {
      buffer_digital[i]=255;
    }
    if(buttonState[i]==false)
    {
       buffer_digital[i]=0;
```

```
    }
  }
  if(is_analog_in[i]==true)
   {
     buffer_digital[i] = analogRead(input[i])/4;
   }
  }

//envoie a hephaistos le buffer
heph.setToHephaistos(buffer_digital, NBR_IN);
}
```

analysons les différentes parties du code

```
#include <Hephaistos.h>

#include <SoftPWM.h>
#include <SoftPWM_timer.h>
```

Ajoute la bibliothèque Héphaïstos et la bibliothèque softPWM pour que toutes les orties puissent être en PWM.

La déclaration des variables globales à adapter par l'utilisateur

Cette partie doit toujours être relue et adapté en fonction du circuit électronique branché à votre arduino.

```
////////////////A adapter par
l'utilisateur//////////////////////////////////////
const int BAUDRATE=9600; //vitesse de transmition sur le port série. Doit
être identique dans la fenêtre de configuration arduino de Héphaistos.
//definition des input
const int NBR_IN=6;  //Nombre de patte en entrées. Doit être identique dans
la fenêtre de configuration arduino de Héphaistos.
int input[NBR_IN]={14,15,16,17,18,19}; //definit les numéros des pins
utilisé en entré
boolean is_pullup[NBR_IN]={false,false,false,true,true,true}; //pour chaque
pins définies plus haut: true pour pouvoir relier la pin à la masse et se
passer de résistances
boolean is_analog_in[NBR_IN]={true,true,true,false,false,false}; //pour
chaque pins définies plus haut: true s'il s'agit d'entrés analogiques

///Definition des sortie
const int NBR_OUT=12; //Nombre de patte en sorties
int output[NBR_OUT]={2,3,4,5,6,7,8,9,10,11,12,13};  //definit les numéros de
pin utilisé en sortie
boolean
is_PWM[NBR_OUT]={true,true,true,true,true,true,true,true,true,true,true,true
```

```
}; //true si la sortie est PWM

//////////////////SUITE DU
CODE//////////////////////////////////////////////////////
```

- Le baudrate est la vitesse de transition sur le port série. Elle doit être identique dans le sketch et dans la fenêtre de configuration arduino d'Héphaïstos.
- NBR_IN permet de définir le nombre de pattes de l'arduino à être configurer en entré.
- NBR_OUT permet de définir le nombre de pattes de l'arduino à être configurer en sortie.
- int input[NBR_IN] est un tableau de valeur. Il va contenir le numéro de chaque patte de l'arduino qui est une entrée. Il doit y avoir autant de valeur que NBR_IN. exemple pour définir les pins 3,8,12 comme des entrés dans l'arduino on va écrire.

```
const int NBR_IN=3;
int input[NBR_IN]={3,8,12};
```

- Même chose pour int output[NBR_OUT] mais concernant les sorties (numériques, analogiques).
- boolean is_pullup[NBR_IN] définit pour chaque entrée si elle est en mode pull-up
- boolean is_analog_in[NBR_IN] définit pour chaque entrée si c'est une entrée analogique
- boolean is_PWM[NBR_OUT] définit pour chaque sortie si c'est une sortie PWM (gradué).

La fonction void setup()

Cette fonction est essentiel au langage Arduino. Elle va être effectuer qu'une fois à chaque démarrage de la carte et permet de configurer celle-ci. Sauf cas exceptionnel, elle n'a pas besoin d'être modifier, les modifications de configuration s'effectuant directement via les variable globale définie plus haut.

La fonction void loop()

C'est la deuxième fonction primordiale du langage Arduino. A partir du moment ou la carte à démarrer, elle va exécuter cette fonction en boucle (d'ou le nom). C'est ici que vous définirez ce que vous voulez que votre carte fasse concrètement. Dans l'exemple donnée, à chaque passage, elle va d'abord vérifier les ordres de communications avec Héphaïstos. Lire les entrées et les envoyer à Héphaistos et enfin écrire les valeurs reçues d'Héphaïstos sur les sortie PWM.