Je vais vous expliquer comment faire une commande électrique proportionnel pour actionner toutes électrovanne hydraulique 12v que l’ on peut trouver sur des matériel agricole ou encore industriel…

Avec une carte arduino nano, une manette de jeu video nintendo wii nunchuk et des cartes de chez citron des MD-10POT

Vous trouverez toutes les information concernant la commande par le nunchuk ici :

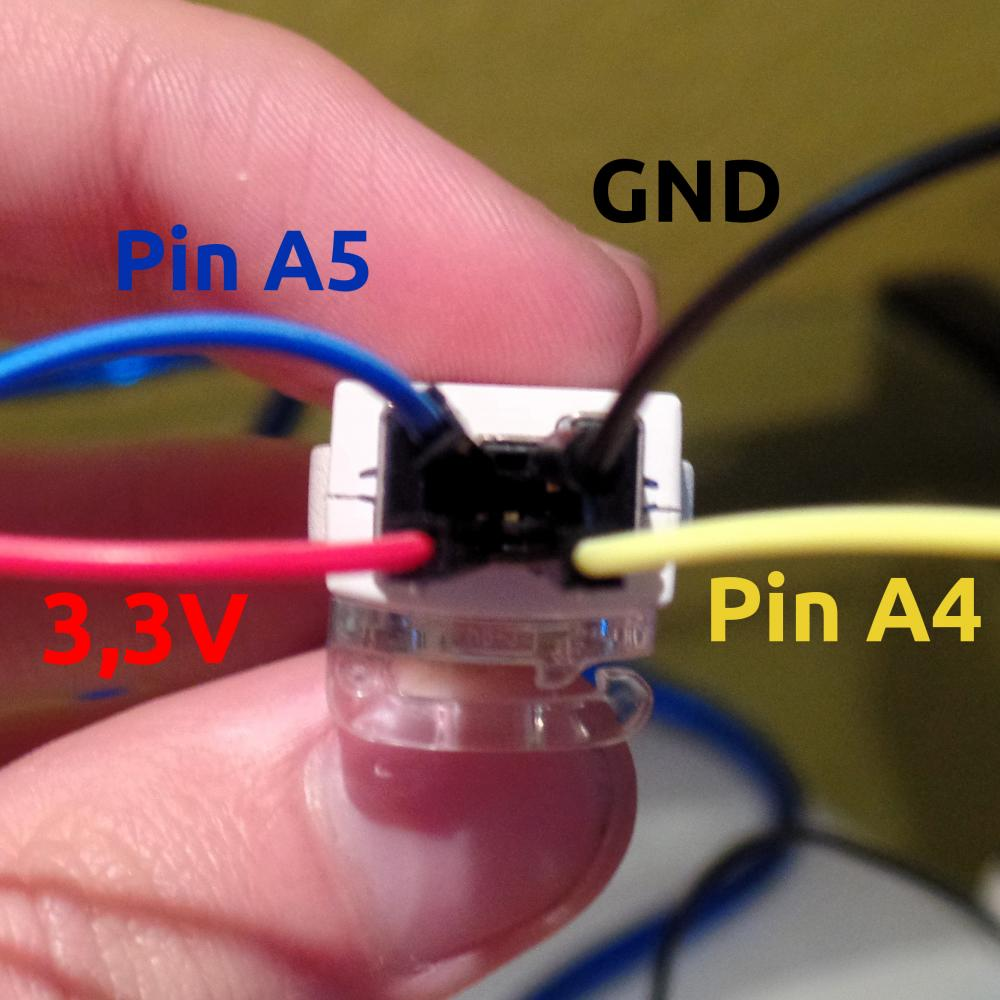
<https://arduino.blaisepascal.fr/nunchuck-et-arduino/>

je vous met une copie partiel avec un commentaire pour vous donner un aperçu :

La manette Nunchuck de Nintendo peut servir comme capteur pour les cartes Arduino, mais il faut utiliser une petite carte d’interface. Ou comme moi coupé les fils et les soudé au support Arduino nano disponible ici [bornier arduino nano](https://www.mouser.fr/ProductDetail/Gravitech/ARD-TERM?qs=Vxac6xGyzPk6DcZhdDk2gw%3D%3D&mgh=1&vip=1&gclid=CjwKCAjw7cGUBhA9EiwArBAvojqhCj8hA3HF0LviR0qKohd1bUqTDSwN6HbwaKEg0VAMv1xu5H75AhoCNtUQAvD_BwE&bm-verify=AAQAAAAF_____1SA6zIwNOk2FQznYAd9MgKiJFMkIQ8aw3Ep5cIvfxX-4Ra-2388p1XPwwdG7VwLlcMBQzWLX_GcszxUInLkym_i7UN2sagxwgct3oc_w3XcMZhx9rp3UVdOVB80BYmt20zxXl9MnZ-4abUcUwE84ahJQtinnApHZoA8YDnQMbusOrjlmypp6C5GNG4YtgIB33EBNufEWHCw0xQ02loKEv7x9ExWumvfAKhzfCUsGOVCs_rzyw324N3FrCf7iFrXVjXUF8JjthVPUFNo63f0TrlGGEGy7TwAFyvRNAVCwXlhQa4u_PfywYkaVk4ZfcPSV4BypZ4MyEBmDuC6Pf53052ZSrJngYWb6Ma9dB4GrjlVcA-Z9ve0G4XmxhUavmnIU9NZ5zXzoijMHd8T9UVjmxkBh7jMW49X6hmwL6y5HTRNFwOo57oUF9Kx4aWnG1M9TnBnHfN-zxnVVt71aepZN9NcfN0e267s8NWQUG2ycu8VjPC5pGKI) . Je préconise soit de soudé directement les fils sur le bornier ou sous le bornier soit de ressoudé le bornier suite au montage car lors du serrage au tournevis les soudure du bornier on tendance a cassé ce qui fait des faut contact qui peuvent être très gênant par la suite.

Vous pouvez donc avec l’ image ci dessous et en démontant la prise repéré la couleur des fils qui sont a branché aux borne de la prise coté nunchuk. 4 fils suffisent a faire fonctionné la nunchuk 3,3v, gnd, a4, et a5.

Vous pouvez retrouvé l’ image avec une explication d’un autre projet sur cette page https://www.robot-maker.com/forum/blog/62/entry-57-utiliser-une-nunchuk-avec-arduino/



Matériel testé : [adaptateur WiiChuck](https://www.gotronic.fr/art-adaptateur-wiichuck-dfr0062-19308.htm)

Bibliothèque nécessaire : [wiichuck](https://github.com/madhephaestus/WiiChuck) (depuis le Library Manager)

## Programmation

Une bibliothèque permet de récupérer facilement les valeurs des divers capteurs (1 joystick, 2 boutons, 2 angles et 1 accéléromètre 3axes). La bibliothèque wiichuck s’installe facilement depuis le [Library Manager de l’IDE Arduino](https://arduino.blaisepascal.fr/installer-une-bibliotheque/).

Pour ce qui est de la carte arduino beaucoup de cour en ligne existe je ne reviendrai pas dessus.

Nous allons utilisé une carte arduino nano et nous aurons besoins d’ utilisé les fonction pwm pour controlé les cartes citrons md10

Passons au code que j’ ai modifié pour faire fonctionner le bras de la débroussailleuse.

C’est un code simple qui utilise une fonction très connue map ;

Cette fonction permet de convertir une donnée sur une échelle de 0 a 1023 par exemple en cette même donnée de 0 a 255 par exemple.

Et la fonction si/sinon encor plus connue, comme son nom la détermine si une action sinon l’ autre chose.

Pour lire ce code commenté il est vraiment préférable de l’ ouvrir dans un ide Arduino afin d’ avoir le code couleur qui permet de visualisé ce qui est commentaire en gris et fonction en noir ou couleurs selon les fonctions, cela n’a pas trop d’ intérêt présenté comme il est ici

//ce code fonctionne, il est bien plus fluide que la version precedante all\_v2\_2 par le commencement des pwm dés la valeur 25

// prochaine etape un manometre pour reduire la valeur forte des pwm en nommant force par exemple

#include <WiiChuck.h>//telecharger la biblioteque pour l' installer

Accessory nunchuck;

//-------------------------------------------------------------------------------

//les pin intensité sont a brancher sur les carte MD10 en pwm

//la pine tete libre/suspenssion est a brancher via un relai 10 amp

//les pin direction sont a brancher sur les carte MD10 en dir

// Ne pas oublié de branché les masses ou gnd des cartes MD10 au gnd de l' arduino

//Pour la nunchuk branché en i2c en 3.3v, gnd et les pin analogique a4 et a5

//---------------

#define pinOuEstBrancheLaLEDr 3 // intensité haut bas La LED (du bras aussi) sera quant à elle branchée sur la sortie D3 de l'Arduino Nano (attention : toutes les sorties ne permettent pas de générer un signal PWM)

#define pinOuEstBrancheLaLEDro 9 // intensité gauche droite La LED (du bras aussi) sera quant à elle branchée sur la sortie D9 de l'Arduino Nano (attention : toutes les sorties ne permettent pas de générer un signal PWM)

#define pinOuEstBrancheLaLEDlibre 11 // la led de l' electrovane tete libre en LOW a verifié

#define pinOuEstBrancheLaLEDp 10 // intensité tete gauche droite La LED ( de la tete aussi) sera quant à elle branchée sur la sortie D9 de l'Arduino Nano (attention : toutes les sorties ne permettent pas de générer un signal PWM)

#define pinOuEstBrancheLaLEDpa 6 // intensité avant arriere La LED ( du bras aussi) sera quant à elle branchée sur la sortie D9 de l'Arduino Nano (attention : toutes les sorties ne permettent pas de générer un signal PWM)

int pinDirPA = 7; // fonction du choix de la direction devant derriere associé a la pine 7

int pinDirP = 12; // fonction du choix de la direction tete gauche droite associé a la pine 1

int pinDirHB = 2; // fonction du choix de la direction gauche associé a la pine 2

int pinDirGD = 4; // fonction du choix de la direction haut bas associé a la pine 4

int valTensionAnalogx; // Variable qui contiendra la valeur de la tension mesurée sur l'entrée analogique haut bas(valeur comprise entre 0 et 1023, car lecture sur 10 bits)

int valTensionAnalogy; //Variable qui contiendra la valeur de la tension mesurée sur l'entrée analogique droite gauche

int valeurSignalPwm1; // Variable qui contiendra la valeur du rapport cyclique du signal PWM à générer pour haut

int valeurSignalPwm2; // Variable qui contiendra la valeur du rapport cyclique du signal PWM à générer pour droite

int valeurSignalPwm3; // Variable qui contiendra la valeur du rapport cyclique du signal PWM à générer pour bas

int valeurSignalPwm4; // Variable qui contiendra la valeur du rapport cyclique du signal PWM à générer pour gauche

int valeurSignalPwm5; // Variable qui contiendra la valeur du rapport cyclique du signal PWM à générer pour pivoté la tete a droite

int valeurSignalPwm6; // Variable qui contiendra la valeur du rapport cyclique du signal PWM à générer pour pivoté la tete a gauche

int valeurSignalPwm7; // Variable qui contiendra la valeur du rapport cyclique du signal PWM à générer pour pivoté pivoté bras vers devant

int valeurSignalPwm8; // Variable qui contiendra la valeur du rapport cyclique du signal PWM à générer pour pivoté pivoté bras vers arriére

void setup() {

delay(1000);

Serial.begin(115200);

Serial.println("Chargement ...");

nunchuck.begin();

if (nunchuck.type == Unknown) {

/\* Si le peripherique n'est pas detecte automatiquement, le nommer explicitement :

NUNCHUCK,

WIICLASSIC,

GuitarHeroController,

GuitarHeroWorldTourDrums,

DrumController,

DrawsomeTablet,

Turntable

\*/

nunchuck.type = NUNCHUCK;

}

pinMode(pinDirHB, OUTPUT); // declare les pin en sortie

pinMode(pinDirPA, OUTPUT);

pinMode(pinDirP, OUTPUT);

pinMode(pinOuEstBrancheLaLEDgenerale, OUTPUT);

pinMode(pinOuEstBrancheLaLEDlibre, OUTPUT);

pinMode(pinOuEstBrancheLaLEDazote, OUTPUT);

}

void loop() {

Serial.println("-------------------------------------------");

nunchuck.readData(); // Lire les entrees et les ecrire dans values[]

Serial.print("JoyX = "); Serial.println(nunchuck.values[0]); //Joystick axe X

Serial.print("JoyY = "); Serial.println(nunchuck.values[1]); //Joystick axe Y

Serial.print("RollAngle = "); Serial.println(nunchuck.values[2]); //Angle de roulis

Serial.print("PitchAngle = "); Serial.println(nunchuck.values[3]); //Angle d'inclinaison

Serial.print("AccX = "); Serial.println(nunchuck.values[4]); //Acceleration axe X

Serial.print("AccY = "); Serial.println(nunchuck.values[5]); //Acceleration axe Y

Serial.print("AccZ = "); Serial.println(nunchuck.values[6]); //Acceleration axe Z

Serial.print("BoutonZ = "); Serial.println(nunchuck.values[10]); //Bouton Z

Serial.print("BoutonC = "); Serial.println(nunchuck.values[11]); //Bouton C

delay(1); //permet de la ractivité en dessous de 2 et de la souplesse en 2 a 20

valeurSignalPwm1 = map(nunchuck.values[0], 125, 255, 20, 200);// map= Conversion tension-> rapport cyclique

//------------------------------------------------------------il faut convertir les mesures 0-122 de la manette en valeur 0-255 pwm pour la carte MD10

if (valeurSignalPwm1 > 28) //si la valeur depasse 30 alors

{

digitalWrite(pinDirHB, HIGH); //met le sens de direction

analogWrite(pinOuEstBrancheLaLEDr, valeurSignalPwm1); //envoie la valeur pwm convertie de la manette vers la carte MD10

}

else //si non

{

analogWrite(pinOuEstBrancheLaLEDr, 0); //ramene la valeur pwm donc l' electrovane a 0

}

valeurSignalPwm2 = map(nunchuck.values[0], 125, 0, 20, 200); // et ainsi de suite comme precedament en inversant les valeur pour faire le mouvement inversé

//---

if (valeurSignalPwm2 > 28)

{

digitalWrite(pinDirHB, LOW);

analogWrite(pinOuEstBrancheLaLEDr, valeurSignalPwm2);

}

//--------------------------------------------------------

valeurSignalPwm3 = map(nunchuck.values[1], 123, 255, 20, 150); // On recommence pour une autre fonction

if (valeurSignalPwm3 > 28)

{

digitalWrite(pinDirGD, HIGH);

analogWrite(pinOuEstBrancheLaLEDro, valeurSignalPwm3);

}

else // on met un sinon juste pour l' electrovane proportionel

{

analogWrite(pinOuEstBrancheLaLEDro, 0);

}

//---

valeurSignalPwm4 = map(nunchuck.values[1], 123, 0, 20, 150);

if (valeurSignalPwm4 > 28)

{

digitalWrite(pinDirGD, LOW);

analogWrite(pinOuEstBrancheLaLEDro, valeurSignalPwm4);

}

//--------------------------------------------------------------------------

valeurSignalPwm5 = map(nunchuck.values[2], 0, 25, 20, 220);

if (nunchuck.values[2] < 99 && nunchuck.values[2] > 5) // on recommence pour une autre fonction (pivoté la tete)

{

digitalWrite(pinDirP, HIGH);

analogWrite(pinOuEstBrancheLaLEDp, valeurSignalPwm5); // on donne la valeur pwm5 de 0 a 255 a pinOuEstBrancheLaLEDp

}

else // on met un sinon juste pour l' electrovane proportionel

{

analogWrite(pinOuEstBrancheLaLEDp, 0);

}

//-----

valeurSignalPwm6 = map(nunchuck.values[2], 255, 233, 20, 220);

if (nunchuck.values[2] < 255 && nunchuck.values[2] > 100) // dans l' autre sens

{

digitalWrite(pinDirP, LOW);

analogWrite(pinOuEstBrancheLaLEDp, valeurSignalPwm6);

}

//-------------------------------------------------------------------------

valeurSignalPwm7 = map(nunchuck.values[3], 0, 15, 0, 255);

if (nunchuck.values[3] < 200 && nunchuck.values[3] > 3) // fonction devant derriere

{

digitalWrite(pinDirPA, HIGH);

analogWrite(pinOuEstBrancheLaLEDpa, valeurSignalPwm7);

}

else // on met un sinon juste pour l' electrovane proportionel

{

analogWrite(pinOuEstBrancheLaLEDpa, 0);

}

//-----

valeurSignalPwm8 = map(nunchuck.values[3], 255, 235, 0, 255);

if (nunchuck.values[3] < 255 && nunchuck.values[3] > 122)

{

digitalWrite(pinDirPA, LOW);

analogWrite(pinOuEstBrancheLaLEDpa, valeurSignalPwm8);

}

//--------------------------------------------------------------------------

if (nunchuck.values[10] > 100) // avant derniére fonction simple electrovane de liberation de pression

{ // pour donner la mobilité a la tete afin de suivre le terrain

digitalWrite(pinOuEstBrancheLaLEDlibre, LOW); // pour donner la mobilité au bras et a la tete afin de suivre le terrain et evité les secousses

}

//--------------------------------------------------------------------------

if (nunchuck.values[11] > 100) // derniére fonction pour arreter l'avant derniére fonction

{

digitalWrite(pinOuEstBrancheLaLEDlibre, HIGH);

}

}