

ChatMagLev

lévitation
magnétique -
version IA



Figure 1. Le matériel du projet MagLev.

Peter Neufeld (Allemagne)

Le nouvel outil précieux des amateurs de programmation est ChatGPT. Il permet non seulement d'écrire un nouveau code, mais aussi d'étendre et d'améliorer un programme existant. Dans cet article, Peter Neufeld, un auteur du magazine Elektor le teste et apporte plusieurs améliorations à son populaire projet de lévitation magnétique.

Quiconque connaît le monde des projets matériels et logiciels sait qu'ils ont tendance à se poursuivre longtemps après leur achèvement. Le processus de réalisation et de programmation peut être agréable, et le projet qui en résulte peut fonctionner bien. Pourtant, il y a souvent des fonctions que vous souhaite-

riez modifier ou ajouter au logiciel, et vous ne semblez jamais trouver le temps, à cause d'autres engagements.

C'était également le cas du code Arduino C++ que j'ai écrit pour mon projet, *lévitation magnétique - version numérique*, que j'ai décrit dans un article d'Elektor [1], sur la plateforme Elektor Labs [2], et sur mon propre site web [3]. Le matériel fonctionne bien, mais j'ai rapidement développé le code Arduino pour montrer que ce projet fonctionne en principe. Cependant, "fonctionnel" ne veut pas dire "parfait !". Lorsque j'ai commencé à m'intéresser aux possibilités de l'IA, je me suis rapidement demandé si ChatGPT [4] pouvait m'aider dans mes projets logiciels de loisir.

Lors de mes premiers tests, j'ai pu très rapidement obtenir de petits extraits de code et des programmes Arduino C++ complets générés par ChatGPT qui correspondaient à mes spécifications formulées dans un langage technique simple.

Par conséquent, deux approches pour améliorer mes projets à l'aide de l'IA se sont concrétisées en fin de compte :

1. Extension de programmes existants : est-il possible d'étendre le code Arduino C++ déjà en cours d'exécution avec des fonctions logicielles supplémentaires en donnant le code existant et en posant des questions spécifiques à ChatGPT ?

2. Nouveau code : ChatGPT peut-il générer un nouveau programme Arduino C++ exécutable si je confie la tâche à l'IA sous la forme d'une description de fonction presque conversationnelle ?

J'ai testé les deux méthodes pour mon "projet de lévitation magnétique" basé sur l'ESP32. La **figure 1** montre la configuration matérielle et la **figure 2** le circuit actuel.

Pour anticiper mon résultat : en tant que programmeur amateur, certaines portes qui n'étaient auparavant qu'à moitié ouvertes, voire complètement fermées, s'ouvrent soudain. Cela fait presque oublier les aspects incontestablement sombres de l'utilisation des systèmes IA. Le système MagLev qui fonctionne bien et que l'on voit sur la photo d'introduction ci-dessus est un résultat visible du travail assisté par l'IA.

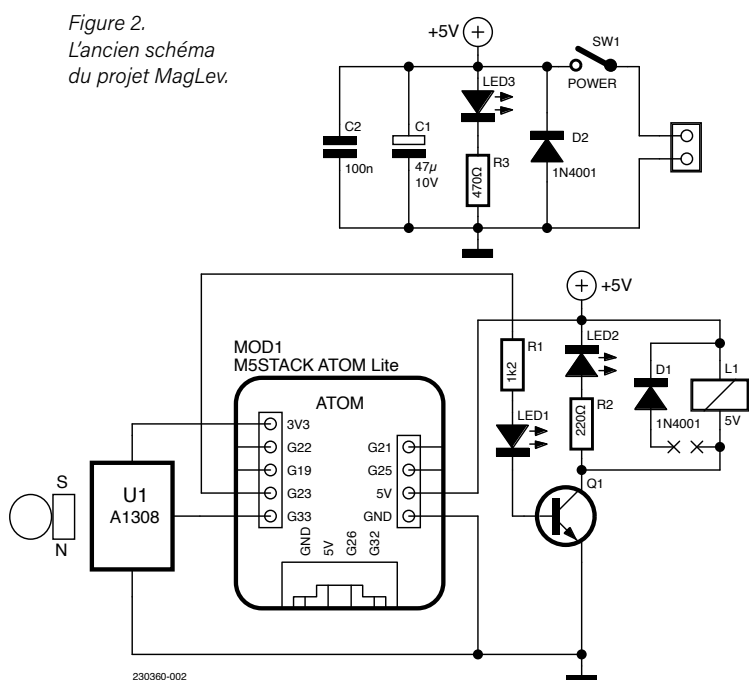
Approche 1 : extensions de l'ancien code du programme

Mon ancien code était fonctionnel, mais n'était que le squelette d'un programme nécessitant des améliorations utiles. J'ai essayé de résumer mes pensées en une seule question et tâche. Mais, généralement, ChatGPT vous permet de développer une conversation, en gérant les questions de suivi.

Voici ma question plus ou moins complexe à ChatGPT :

Le code Arduino C++ suivant pour une ESP32 doit être complété. Les valeurs de déclenchement doivent maintenant être augmentées ou diminuées de 1 avec le caractère "T" ou "t" suivi de CR et LF sur la ligne série. Respectivement pour la valeur d'hystérésis. Une quatrième commande stocke toutes les valeurs actuelles dans la RAM. Elles sont utilisées au démarrage si elles sont présentes. Utilisez la bibliothèque preferencec. Les nouvelles valeurs sont affichées en série. Ajoutez le code de programme nécessaire et donnez le programme complet.

Figure 2.
L'ancien schéma
du projet MagLev.



```

/*****

```

Magnetic Levitation object:

Lets a LEGO man, glued together with a neobodymium magnet, float under a modified 5V relay

- SoC: ESP32, very good:

M5Stack's ATOM LITE or ATOM MATRIX

- Electromagnet: Modified 5V-Relais HW-482 with 3V3/5V transistor input

- Sensor: HALL-sensor A1302 or A1308

```

*****/

```

```

// int TRIGGER = 2740;
// Triggerlevel is set to a level where
// weight of payload is equal to the force
// between magnet and electromagnet
int TRIGGER = 2740;
// good for payload =
// 2 Neobdym-Magnets and a LEGO-Man
int HYST = 35;
// Hysteresis for trigger level
int HALL_PIN = 33;
// analog Signal from HALL-sensor at GPIO33
int HALL_VAL = 0;
int RELAIS_PIN = 23;
// GPIO23 to drive the transistor input of
// the modified 5V-relais that is used as electromagnet
int X = 0;

```

```

void setup(){
  Serial.begin(115200);
  pinMode(RELAIS_PIN, OUTPUT);
  Serial.print("Magnetic Levitation: START ");
}

```

```

void loop(){
  HALL_VAL = analogRead(HALL_PIN);
  //read HALL-Sensor with
  //default 0-3.9V input >> 12bit
  if (HALL_VAL < (TRIGGER + X)){
    digitalWrite(RELAIS_PIN, HIGH);
    // lift the payload
    X = HYST;
  }
  else{
    digitalWrite(RELAIS_PIN, LOW);
    // drop the payload
    X = 0 - HYST;
  }
}

```

Vous pouvez voir mon ancien code et la réponse de ChatGPT avec le code amélioré dans le **listage 1**.



Listage 1. Extension of handwritten code.

Voici le code étendu du programme Arduino C++ que vous avez fourni :

```
#include <Preferences.h>
Preferences preferences;

int TRIGGER = 2740;
int HYST = 35;
int HALL_PIN = 33;
int HALL_VAL = 0;
int RELAIS_PIN = 23;
int X = 0;

void setup() {

    Serial.begin(115200);
    pinMode(RELAIS_PIN, OUTPUT);
    Serial.print("Magnetic Levitation: START ");

    // Load values from non-volatile RAM
    // Initialize preferences with a namespace
    preferences.begin("levitation", false);

    if (preferences.getBytesLength("config") > 0) {
        preferences.getBytes("config", &TRIGGER,
                               sizeof(TRIGGER));
        preferences.getBytes("config", &HYST,
                               sizeof(HYST));
    }

    preferences.end();
}

void loop() {

    if (Serial.available() >= 3) {
        String command = Serial.readStringUntil('\r');
        // Read the command until a
        // carriage return (\r) is encountered

        if (command.startsWith("T")) {
            TRIGGER++;
            Serial.println("Trigger increased: "
                           + String(TRIGGER));
        }
        else if (command.startsWith("t")) {
            TRIGGER--;
            Serial.println("Trigger decreased: "
                           + String(TRIGGER));
        }
        else if (command.startsWith("H")) {
            HYST++;
            Serial.println("Hysteresis increased: "
```

```
        + String(HYST));
    }
    else if (command.startsWith("h")) {
        HYST--;
        Serial.println("Hysteresis decreased: "
                       + String(HYST));
    }
    else if (command.startsWith("S")) {
        // Store current values in non-volatile RAM
        // Initialize preferences with a namespace
        preferences.begin("levitation", false);
        preferences.putBytes("config", &TRIGGER,
                               sizeof(TRIGGER));
        preferences.putBytes("config", &HYST,
                               sizeof(HYST));
        preferences.end();
        Serial.println(
            "Values stored in non-volatile RAM.");
    }
}

HALL_VAL = analogRead(HALL_PIN);

if (HALL_VAL < (TRIGGER + X)) {
    digitalWrite(RELAIS_PIN, HIGH);
    X = HYST;
} else {
    digitalWrite(RELAIS_PIN, LOW);
    X = 0 - HYST;
}
}
```

In this code, the preferences library is used to store and load the trigger and hysteresis values in non-volatile RAM. The new commands 'T', 't', 'H', and 'h' are used to increase the trigger, decrease the trigger, increase the hysteresis, and decrease the hysteresis values, respectively. The command 'S' stores the current values in non-volatile RAM. These values will be used at startup if present. Please note that you need to install the "Preferences" library for this code to work correctly. You can find the library on the Arduino Library Manager or download it from the Arduino website.

Ma première impression : ouah ! C'était facile ! Le code semble correct et logique. Il répond à mes exigences et fonctionne très bien lorsque je le teste ! J'aurais pu l'écrire de la même manière, mais seulement si j'étais plus expérimenté en programmation C++ et/ou si j'avais cherché et adapté d'autres exemples de code – et bien sûr, y consacrer du temps.

Laissons de côté l'ancien code et reprenons depuis le début avec quelques phrases claires.

Approche 2 : un nouveau code Arduino C++

Motivée, je me suis alors lancée dans une description conversationnelle de mon montage de lévitation nécessitant des améliorations et plus de fonctions logicielles... et j'ai été très étonnée de la qualité du résultat. Voici ma conversation avec ChatGPT :

Le système de lévitation magnétique suivant doit être contrôlé par un ESP32 et un programme Arduino C++. La broche 23 contrôle un électroaimant avec un noyau de fer, la broche 33 est l'entrée CAN à laquelle un capteur à effet Hall analogique est connecté. Le capteur analogique à effet Hall se trouve sous l'électroaimant. Un aimant permanent s'approche de l'électroaimant et de son noyau de fer, ainsi que du capteur de Hall. L'aimant est attiré par le noyau. Il doit cependant être maintenu en suspension. Pour ce faire, l'électroaimant doit être brièvement mis hors tension.

Le capteur à effet Hall génère une tension d'environ 2,4 V à l'état flottant et d'environ 1,5 V sans l'aimant permanent. Si l'aimant s'approche du capteur, la tension augmente jusqu'à un maximum de 3 V.

Pendant l'exécution du programme, la valeur de déclenchement pour l'état flottant doit être légèrement augmentée via l'interface série (115200Baud, 8N1) à la réception d'un "T". À la réception d'un "t", elle est diminuée.

De plus, comme la valeur de déclenchement, une valeur d'hystérésis d'environ 50 mV doit également être réglée et corrigée en conséquence par des commandes au clavier.

Les valeurs de déclenchement et d'hystérésis corrigées manuellement doivent être éditées si elles ont été modifiées. Les deux valeurs doivent être enregistrées en toute sécurité dans la mémoire non volatile de l'ESP32 lors de la saisie de "S" ou "s". Ce n'est que s'il existe déjà d'anciennes valeurs dans la mémoire non volatile qu'elles doivent être utilisées comme valeurs initiales du programme. Une brève description de l'opération doit être affichée au démarrage. Écrivez le programme Arduino de manière très compacte et avec un nombre raisonnablement réduit de commentaires, de sorte qu'il puisse être édité entièrement dans la réponse.

Dans le **listage 2**, vous pouvez voir la réponse de ChatGPT, qui m'étonne encore et me fait même un peu peur.



Listage 2. Code et réponse générés par ChatGPT.

```
#include <EEPROM.h>

const int magnetPin = 23;
const int hallSensorPin = 33;
const int triggerAddr = 0;
const int hysteresisAddr = 1;

float triggerValue = 2.4;
float hysteresisValue = 0.05;
```

```
void setup() {
  Serial.begin(115200);
  pinMode(magnetPin, OUTPUT);
  pinMode(hallSensorPin, INPUT);

  // Load trigger and hysteresis values
  // from EEPROM if available
  EEPROM.begin(2);
  float storedTriggerValue =
    EEPROM.readFloat(triggerAddr);
  float storedHysteresisValue =
    EEPROM.readFloat(hysteresisAddr);

  if (storedTriggerValue != 0x00)
    triggerValue = storedTriggerValue;

  if (storedHysteresisValue != 0x00)
    hysteresisValue = storedHysteresisValue;

  Serial.println("Magnetic Levitation Setup");
  Serial.println("Commands:");
  Serial.println("T - Increase trigger value");
  Serial.println("t - Decrease trigger value");
  Serial.println("H - Increase hysteresis value");
  Serial.println("h - Decrease hysteresis value");
  Serial.println(
    "S or s - Save trigger and hysteresis values");
  Serial.println();
  Serial.print("Trigger Value: ");
  Serial.println(triggerValue);
  Serial.print("Hysteresis Value: ");
  Serial.println(hysteresisValue);
}

void loop() {
  // Read the analog value from the Hall sensor
  int hallSensorValue = analogRead(hallSensorPin);

  // Check if magnet is close to the sensor
  if (hallSensorValue >
    (triggerValue + hysteresisValue) * 1023) {
    // Magnet is close, turn off the electromagnet
    digitalWrite(magnetPin, LOW);
  }
  else {
    // Magnet is far, turn on the electromagnet
    digitalWrite(magnetPin, HIGH);
  }

  // Check for serial input
  if (Serial.available() > 0) {
    char command = Serial.read();
    if (command == 'T') {
      triggerValue += 0.01;
    }
  }
}
```

```

else if (command == 't') {
    triggerValue -= 0.01;
}
else if (command == 'H') {
    hysteresisValue += 0.01;
}
else if (command == 'h') {
    hysteresisValue -= 0.01;
}
else if (command == 'S' || command == 's') {
    // Save trigger and hysteresis values
    // to EEPROM
    EEPROM.writeFloat(triggerAddr, triggerValue);
    EEPROM.writeFloat(hysteresisAddr,
                      hysteresisValue);
    EEPROM.commit();
    Serial.println(
        "Trigger and hysteresis values saved.");
}

// Output trigger and hysteresis values
Serial.print("Trigger Value: ");
Serial.println(triggerValue);
Serial.print("Hysteresis Value: ");
Serial.println(hysteresisValue);
}
}

```

Note: This is a compact version of the code without extensive comments. It assumes you have the necessary libraries installed, including the EEPROM library. Make sure to include that library in your Arduino IDE if you don't have it already.

Ensuite, j'ai demandé au bot de donner une brève description du code à des fins de documentation.

ChatGPT a répondu :

Le code fourni est un programme Arduino compact écrit en C++ pour contrôler un circuit de lévitation magnétique avec un micro-contrôleur ESP32. La configuration comprend un électroaimant avec un noyau en fer et un capteur à effet Hall analogique. Au démarrage, le programme initialise la communication série avec un débit de bauds de 115 200. La broche 23 est utilisée pour commander l'électroaimant, et la broche 33 est configurée en tant qu'entrée analogique pour lire la tension de sortie du capteur à effet Hall. Le programme récupère les valeurs de déclenchement et d'hystérésis de la mémoire non volatile (EEPROM) s'ils ont été précédemment enregistrés. Si aucune valeur n'est trouvée, il utilise des valeurs par défaut. La valeur de déclenchement représente le seuil de tension pour éteindre l'électroaimant, et la valeur d'hystérésis définit la plage de tension dans laquelle l'électroaimant reste éteint même si l'aimant est légèrement éloigné du capteur. Le programme affiche ensuite une brève description de l'opération et des commandes disponibles sur le moniteur série. Les commandes permettent d'ajuster les valeurs de déclenchement et d'hystérésis

en les incrémentant ou en les décrémentant. Les valeurs mises à jour sont immédiatement affichées sur le moniteur série. Pour enregistrer les valeurs de déclenchement et d'hystérésis dans la mémoire non volatile, l'utilisateur peut entrer la commande "S" ou "s". Dans la boucle main, le programme lit la valeur analogique du capteur de Hall et la compare à la valeur de déclenchement plus la valeur d'hystérésis. Si la lecture du capteur est supérieure à ce seuil, ce qui indique que l'aimant est proche, l'électroaimant est éteint en mettant la broche 23 à un niveau bas. Sinon, si l'aimant est éloigné, l'électroaimant est activé en mettant la broche 23 à un niveau haut.

Le programme vérifie continuellement les commandes reçues depuis le moniteur série. Si une commande est reçue, il ajuste la valeur de déclenchement ou d'hystérésis en conséquence. Si la commande "S" ou "s" est entrée, les valeurs de déclenchement et d'hystérésis sont enregistrées dans la mémoire non volatile.

Les valeurs actuelles de déclenchement et d'hystérésis sont affichées sur le moniteur série chaque fois qu'elles sont modifiées ou au démarrage du programme.

Ce programme fournit une solution simple et compacte pour contrôler une configuration de lévitation magnétique à l'aide d'un micro-contrôleur ESP32.

Amélioration

Jusqu'à présent, tout va bien, mais tout peut toujours être optimisé. J'ai demandé au bot de me donner trois brèves suggestions pour améliorer la boucle de régulation.

ChatGPT m'a donné ces suggestions en retour :

1. *Implémentez une régulation PID : considérez l'implémentation d'une boucle de régulation PID (Proportionnel-Intégral-Dérivé) pour améliorer la stabilité et la réactivité du système de lévitation magnétique. En utilisant la rétroaction du capteur de Hall et en appliquant les gains de contrôle appropriés, le régulateur PID peut ajuster dynamiquement la puissance de l'électroaimant pour maintenir la distance de lévitation souhaitée. Cela améliorera la capacité du système à gérer les perturbations et à fournir un contrôle plus fluide.*
2. *Utilisez un filtre de moyenne mobile : pour réduire le bruit et les fluctuations dans les lectures du capteur à effet Hall, appliquez un filtre de moyenne mobile. Au lieu de vous fier uniquement aux lectures individuelles, calculez la valeur moyenne sur un petit intervalle de lectures récentes. Cela contribuera à fournir une mesure plus stable et fiable pour déterminer la position de l'aimant et ajuster en conséquence la puissance de l'électroaimant.*
3. *Intégrez une fonction de délai de sécurité : ajoutez un mécanisme de délai de sécurité à la boucle de régulation pour vous assurer que l'électroaimant n'est pas alimenté en continu si l'aimant ou le capteur est absent ou en panne. Si le système ne reçoit pas une lecture valide du capteur de Hall dans un délai spécifié, il peut supposer qu'il y a un problème et prendre les mesures appropriées, telles que l'extinction de l'électroaimant ou le déclenchement d'une condition d'erreur.*

En implémentant ces suggestions, vous pouvez améliorer les performances de la boucle de régulation, l'immunité au bruit et les fonctionnalités de sécurité, ce qui se traduira par un contrôle de lévitation magnétique plus précis et fiable.

J'admets que j'ai posé mes questions initiales après de longues discussions au cours desquelles j'ai dû apprendre à formuler mes questions d'une manière adaptée à l'IA de ChatGPT.

D'ailleurs, les réponses de ChatGPT étaient souvent interrompues brusquement. Lorsque je lui ai demandé de limiter la longueur de sa réponse, il m'a répondu par un "Oui plus 2048 caractères". C'est pourquoi j'ai également inclus dans ma requête des indices permettant de raccourcir la réponse de manière appropriée.

Toutefois, si l'on s'est un peu familiarisés avec les réactions du système, les résultats sont également très bons après avec des questions, des exigences et des instructions un peu plus complexes. Des demandes ciblées permettent d'ajouter d'autres fonctions plus détaillées.

Si le résultat semble incorrect d'un point de vue technique, il est intéressant de constater que la même question répétée plusieurs fois reçoit une réponse différente à chaque fois, ce qui se traduit par un code dont la structure est parfois complètement différente, mais qui est le plus souvent fonctionnel. Il est alors très utile, voire indispensable, d'au moins bien lire le code généré et de vérifier que toutes les exigences sont respectées. ChatGPT a vite fait d'omettre l'un ou l'autre aspect ! Vous pouvez remettre en question les suggestions de l'IA qui contredisent votre propre expertise technique et vos connaissances. Cela peut aussi vous permettre d'acquérir de nouvelles connaissances techniques. En résumé, l'utilisation de ChatGPT pour créer un programme basé sur la description d'un problème en langage naturel s'est avérée très utile dans ce cas. Il s'agit d'un outil réglable et contrôlable.

Même un développeur C++ peu expérimenté comme moi peut maintenant créer rapidement des parties de programmes plus complexes qui m'auraient pris beaucoup plus de temps auparavant. La documentation est désormais très simple. On peut même s'attendre à une sorte d'approfondissement du problème posé, qui ouvre l'horizon à des domaines relatifs inattendus, techniquement bien fondés et détaillés. Cependant : actuellement, l'IA, en particulier ChatGPT, est un outil utile, mais en aucun cas infaillible, pour le développeur amateur, et il a toujours besoin d'une intervention humaine ! Je suis très enthousiaste à l'idée de réaliser d'autres projets avec ce nouveau jeu de tournevis dans ma boîte à outils.

Résoudre certains problèmes

Comme si mes conclusions avaient besoin d'une preuve supplémentaire : en testant le code généré, j'ai remarqué qu'il y avait deux problèmes dans le code. OK... demandons au service ce qu'il en pense !

Corrigez deux problèmes.

1. *La valeur du capteur à effet Hall est toujours comparée à une valeur fixe sans hystérésis.*
2. *Ce code ne stocke pas correctement les valeurs dans le nv-ram de mon ESP32. Utilisez une autre méthode pour le stockage et la lecture.*

Réécrivez le code ci-dessus de manière très compacte et en réduisant le nombre de commentaires, afin qu'il puisse être édité intégralement dans la réponse.

ChatGPT a répondu "Apologies for the oversight" (!) et m'a renvoyé le code du **listage 3**.



Listage 3. Code amélioré.

```
#include <Preferences.h>

const int magnetPin = 23;
const int hallSensorPin = 33;
const char* prefNamespace = "levitation";
const char* triggerKey = "trigger";
const char* hysteresisKey = "hysteresis";

float triggerValue = 2.4;
float hysteresisValue = 0.05;

Preferences preferences;

void setup() {
    Serial.begin(115200);
    pinMode(magnetPin, OUTPUT);
    pinMode(hallSensorPin, INPUT);

    preferences.begin(prefNamespace, false);
    if (preferences.getBool("initialized", false)) {
        triggerValue =
            preferences.
getFloat(triggerKey, triggerValue);
        hysteresisValue =
            preferences.getFloat(hysteresisKey,
                                hysteresisValue);
    }
    else {
        preferences.putBool("initialized", true);
    }

    Serial.println("Magnetic Levitation Setup");
    Serial.println("Commands:");
    Serial.println("T - Increase trigger value");
    Serial.println("t - Decrease trigger value");
    Serial.println("H - Increase hysteresis value");
    Serial.println("h - Decrease hysteresis value");
    Serial.println(
        "S or s - Save trigger and hysteresis values");
    Serial.println();
    Serial.print("Trigger Value: ");
    Serial.println(triggerValue);
    Serial.print("Hysteresis Value: ");
    Serial.println(hysteresisValue);
}

void loop() {
    int hallSensorValue = analogRead(hallSensorPin);

    if (hallSensorValue > (triggerValue * 1023) &&
        digitalRead(magnetPin) == HIGH) {
        digitalWrite(magnetPin, LOW);
    }
}
```

```

else if (hallSensorValue <
        ((triggerValue - hysteresisValue) *
         1023) && digitalRead(magnetPin) == LOW)
{
    digitalWrite(magnetPin, HIGH);
}

if (Serial.available() > 0) {
    char command = Serial.read();
    if (command == 'T') {
        triggerValue += 0.01;
    } else if (command == 't') {
        triggerValue -= 0.01;
    } else if (command == 'H') {
        hysteresisValue += 0.01;
    } else if (command == 'h') {
        hysteresisValue -= 0.01;
    } else if (command == 'S' || command == 's') {
        preferences.putFloat(triggerKey,
                             triggerValue);
        preferences.putFloat(hysteresisKey,
                             hysteresisValue);
        preferences.end();
        Serial.println(
            "Trigger and hysteresis values saved.");
    }

    Serial.print("Trigger Value: ");
    Serial.println(triggerValue);
    Serial.print("Hysteresis Value: ");
    Serial.println(hysteresisValue);
}
}

```

This version addresses the two issues mentioned:

The code now includes hysteresis in the comparison by checking the magnetPin state in both conditions. If the electromagnet is already in the desired state, the condition for turning it on or off is not evaluated.

Instead of using the EEPROM library, this code uses the Preferences library provided by the ESP32 core. The preferences are stored in the non-volatile memory of the ESP32 using the Preferences API. The values are saved when the 'S' or 's' command is received and are read from the memory during setup. The preferences namespace and keys are specified for the levitation setup to avoid conflicts with other preferences.

Eh bien, cela fonctionne très bien maintenant. Pendant les tests, cependant, j'ai découvert un autre petit problème ! Mais maintenant, dans ce stade amélioré, j'en suis venu à la conclusion que ce projet de logiciel devrait également rester inachevé. J'ai donc mis fin à la discussion et j'ai pris un café avec ma femme. ◀

230360-04

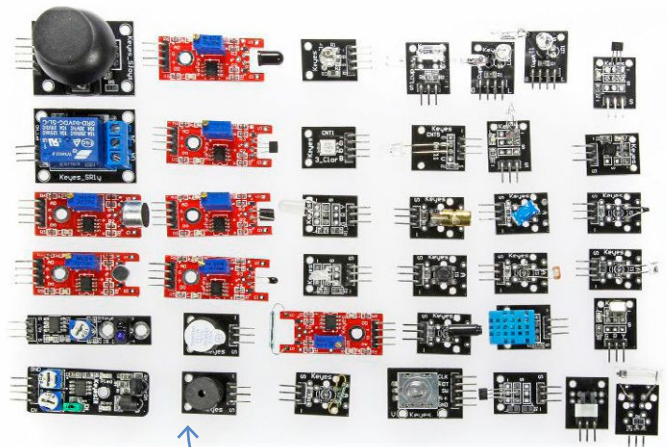


À propos de l'auteur

Pendant plus de 40 ans, Peter Neufeld a été chargé de superviser les systèmes informatiques, électriques, de contrôle des bâtiments, et des systèmes de communication d'autres personnes. Aujourd'hui, il consacre du temps à réaliser des applications pratiques de l'électronique classique et des projets avec des microcontrôleurs modernes.. Il a décrit certains de ces projets sur la plateforme Elektor Labs ou sur son blog.

Questions ou commentaires ?

Contactez Elektor (redaction@elektor.fr).

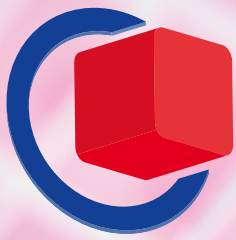


Produit

> 37-en-1 capteurs d'Elektor
www.elektor.fr/16843

LIENS

- [1] Peter Neufeld, "Lévitiation magnétique, version numérique" Elektor 9-10/2021 : <https://www.elektormagazine.fr/magazine/elektor-184/59887>
- [2] "Magnetic Levitation - The Digital Way," Elektor Labs : <https://elektormagazine.fr/labs/magnetic-levitation-the-digital-way>
- [3] MagLev sur PeterNeufeld.wordpress.com : <https://peterneufeld.wordpress.com/category/magnetic-levitation>
- [4] ChatGPT : <https://chat.openai.com>



embeddedworld

Exhibition & Conference



CONNECTING THE
EMBEDDED COMMUNITY

9 – 11.4.2024



Get your
free ticket now!

embedded-world.de/codes

Use the voucher code **ew24ELE**

Media partners

Markt & Technik
die unverzichtbare Wochenzeitschrift für Elektronik

Elektronik

**computer &
automation**

Elektronik
automotive

Elektronik
•medical

elektroniknet.de

NÜRNBERG MESSE