

Choix :

- Carte électronique
- Langage de programmation
- Caméra thermique

Partie chaima :






1) Carte Électronique :

Nous avons effectué une étude pour identifier la carte électronique la plus adaptée à nos besoins en vue de réaliser notre projet. Lors de la comparaison de différentes cartes électroniques, il est important de prendre en compte plusieurs critères clés.

Nous pouvons notamment citer la puissance de traitement qui est cruciale pour gérer des tâches complexes et traiter de grandes quantités de données, la capacité de mémoire pour le stockage et l'accès aux données, la connectivité importante pour établir des communications et transférer des données, ainsi que les interfaces et les E/S disponibles sur la carte électronique pour l'interaction avec d'autres périphériques ou capteurs. Enfin, l'alimentation est également un facteur essentiel pour un fonctionnement fiable, que ce soit pour une utilisation sur secteur ou avec des batteries.

En évaluant ces critères, il est possible de comparer efficacement les différentes cartes électroniques et de trouver celle qui correspond le mieux aux besoins spécifiques de chaque application.

→ Le tableau montre une comparaison de 5 cartes électroniques distinctes:

Spécifications	ESP8266 CP2102 NodeMCU LUA 	ESP32- DEVKIT WROOM-32 	Raspberry Pi Pico W 	Raspberry Pi Zero WH 	Raspberry Pi 4 
Alimentation (V)	3.3	3.3	1.5-5.5	5	5V via USB-C
Processeur	Xtensa LX106	Tensilica LX6	Arm Cortex-M0+	ARM11	ARM Cortex-A72
Nombre de coeurs	1	2	2	1	4
Mémoire flash	4 MB	4 Mo	2 MB	512 MB	Carte microSD 8 Go à 256 Go
Mémoire SRAM	520 kB	520 ko	264 Ko	SDRAM 512 Mo	SDRAM (2/4/8) Go
Fréquence (MHz)	80 à 240	80 à 240	133	1000	1500
Connexion	Wifi	Wifi / Bluetooth	Wifi	Wifi / Bluetooth	Wifi / Bluetooth

Nombre de broches	17	30	40	40	40
Coût (TND)	25.00	40.00	43.00	145,00	200-300
Dimensions (mm)	49x24.5x 13	51x26x14	51x21x3.9	65x30x5	85.6x56.5x17
Poids (kg)	0,015	0,015	0,003	0,02	0.046
Langage de programmation	C/C++, Micropython, NodeMCU-Lua	C/C++, Micropython, NodeMCU-Lua	Micropython, C/C++	Micropython, C/C++, Java, JavaScript	Micropython, C/C++ , Java ,JavaScript

- ❖ Après avoir mené une étude pour identifier la carte électronique la plus adaptée à notre projet, nous avons opté pour la carte Raspberry Pi 4. Cette carte répondait à nos besoins spécifiques grâce à sa grande puissance de traitement et sa mémoire généreuse. Elle dispose de 4 cœurs, prend en charge la connexion sans fil via le Wi-Fi et le Bluetooth, offre un nombre de broches suffisant pour notre projet .

2) Langage de programmation :

- Pour réaliser notre projet de Robot Grimpeur Détecteur De Fissures, nous avons étudié les langages de programmation les plus adaptés à notre carte Raspberry Pi 4. Lors de la comparaison des différents langages, il est crucial de prendre en compte plusieurs critères clés.
- Nous pouvons notamment citer la performance du langage qui est essentielle pour gérer des tâches complexes et traiter de grandes quantités de données, la facilité d'utilisation pour le développement et le débogage, la disponibilité des bibliothèques et des frameworks spécifiques aux besoins de notre projet, ainsi que la communauté et le support disponibles pour résoudre les problèmes éventuels. Enfin, la compatibilité avec les capteurs et périphériques est également un facteur essentiel pour une intégration fluide et efficace.
- En évaluant ces critères, il est possible de comparer efficacement les différents langages de programmation et de choisir celui qui correspond le mieux aux besoins spécifiques de notre application. Voici une analyse des caractéristiques des principaux langages de programmation pour la Raspberry Pi 4 :

Critère	C	C++	Python	Java	JavaScript
Performance	Très élevée	Très élevée	Moyenne	Élevée	Moyenne
Facilité d'utilisation	Moyenne	Moyenne	Très élevée	Moyenne	Élevée
Bibliothèques / Frameworks	Large disponibilité	Large disponibilité	Immense disponibilité	Large disponibilité	Large disponibilité
Communauté / Support	Large	Large	Très large	Large	Très large
Compatibilité	Excellente	Excellente	Excellente	Bonne	Bonne

Développement rapide	Faible	Faible	Très rapide	Rapide	Très rapide
Gestion de la mémoire	Manuelle	Manuelle	Automatique	Automatique	Automatique
Sécurité	Variable	Variable	Variable	Élevée	Variable
Évolutivité	Bonne	Bonne	Bonne	Bonne	Bonne
Écosystème	Moyen	Moyen	Très développé	Développé	Développé
Portabilité	Variable	Variable	Élevée	Élevée	Élevée
Interopérabilité	Excellente	Excellente	Bonne	Bonne	Bonne
Documentation	Abondante	Abondante	Abondante	Abondante	Abondante
Maintenance	Nécessite plus de travail	Nécessite plus de travail	Facile	Facile	Facile
Popularité	Très populaire	Très populaire	Très populaire	Populaire	Populaire
Scalabilité	Excellente	Excellente	Bonne	Bonne	Bonne
Simplicité de la syntaxe	Complexité syntaxique	Complexité syntaxique	Syntaxe simple	Complexité syntaxique	Syntaxe simple
Community Trend	Stable	Stable	En croissance	Stable	En croissance

→ Pour notre projet de **Robot Grimpeur Détecteur De Fissures**, Python semble être le choix le plus adapté en raison de sa facilité d'utilisation, de la richesse de ses bibliothèques, et de sa grande communauté de support. Cependant, pour des tâches critiques nécessitant des performances élevées, l'utilisation de C ou C++ pourrait être envisagée, en particulier pour des composants spécifiques où la performance est cruciale.

Pourquoi Python est le Meilleur Choix pour le Projet Robot Grimpeur Détecteur de Fissures ??????

Pour notre projet de Robot Grimpeur Détecteur de Fissures, Python se distingue comme le meilleur choix de langage de programmation pour plusieurs raisons :

1. Facilité d'utilisation et Développement Rapide :

- Python est réputé pour sa syntaxe simple et claire, ce qui permet de réduire le temps de développement et de se concentrer sur la logique du projet plutôt que sur les détails de la syntaxe.
- Le développement rapide est essentiel pour un projet complexe comme un robot grimpeur, où des itérations fréquentes sont nécessaires pour ajuster et améliorer les fonctionnalités.

2. Large Collection de Bibliothèques et de Frameworks :

- Python possède une immense collection de bibliothèques et de frameworks qui simplifient l'intégration de diverses fonctionnalités. Par exemple, pour le

traitement d'image et l'analyse de fissures, des bibliothèques comme OpenCV peuvent être utilisées.

- Pour les aspects de machine learning ou d'intelligence artificielle nécessaires pour l'analyse autonome des fissures, des bibliothèques comme TensorFlow ou PyTorch sont disponibles.

3. Support et Communauté Étendue :

- Python bénéficie d'une très large communauté de développeurs, ce qui signifie qu'une multitude de ressources, de tutoriels et de forums sont disponibles pour aider à résoudre les problèmes rencontrés.
- Un support communautaire fort est crucial pour des projets innovants où des problèmes uniques peuvent surgir.

4. Compatibilité et Portabilité :

- Python est hautement compatible avec le matériel de la Raspberry Pi et prend en charge une grande variété de capteurs et de périphériques via des bibliothèques spécifiques.
- La portabilité de Python permet de développer et tester le code sur différentes plates-formes (Windows, Mac, Linux) avant de le déployer sur la Raspberry Pi.

5. Gestion Automatique de la Mémoire :

- La gestion automatique de la mémoire en Python (via le garbage collection) permet de minimiser les risques de fuites de mémoire et autres erreurs de gestion de mémoire, ce qui simplifie le développement et la maintenance du projet.

6. Écosystème Développé :

- Python possède un écosystème très développé avec des outils de développement puissants (comme Jupyter Notebooks, IDEs comme PyCharm, etc.) qui facilitent le développement, le débogage et le déploiement du code.
- Les outils de visualisation de données comme Matplotlib ou Seaborn peuvent être utilisés pour analyser les données recueillies par le robot.

7. Flexibilité pour Prototypage et Production :

- Python est idéal pour le prototypage rapide, permettant de tester des concepts et des fonctionnalités rapidement.
- Une fois le prototype validé, Python est également suffisamment robuste pour être utilisé en production, ce qui réduit la nécessité de réécrire le code dans un autre langage.

- En combinant facilité d'utilisation, richesse des bibliothèques, support communautaire, compatibilité matérielle, gestion automatique de la mémoire, et un écosystème développé, <<Python>> s'avère être le choix le plus pragmatique et efficace pour notre projet de Robot Grimpeur Détecteur de Fissures. Il nous permet de développer rapidement et efficacement, tout en garantissant que nous pouvons facilement adapter et améliorer notre robot à mesure que le projet progresse.

3) Caméra thermique :

Pourquoi Choisir une Caméra Thermique pour le Projet Robot Grimpeur Détecteur de Fissures ????

L'utilisation d'une caméra thermique dans notre projet de Robot Grimpeur Détecteur de Fissures est justifiée par plusieurs avantages et capacités uniques qu'elle offre par rapport aux autres types de caméras. Voici les raisons principales pour ce choix :

1. Détection des Anomalies Invisibles à l'Oeil Nu :

- Les caméras thermiques peuvent détecter des variations de température à la surface des matériaux, révélant des fissures ou des défauts structurels qui ne sont pas visibles à l'œil nu ou avec des caméras ordinaires.
- Cela permet d'identifier des problèmes avant qu'ils ne deviennent visibles ou plus graves, offrant une prévention proactive des défaillances structurelles.

2. Capacité de Fonctionner dans des Conditions de Faible Lumière :

- Les caméras thermiques ne dépendent pas de la lumière visible pour capturer des images. Elles peuvent fonctionner efficacement dans l'obscurité totale, les environnements mal éclairés ou les zones avec des conditions d'éclairage changeantes.
- Cela est particulièrement utile pour les inspections en extérieur pendant la nuit ou dans des zones intérieures mal éclairées.

3. Analyse de la Distribution de Chaleur :

- Les caméras thermiques permettent de visualiser la distribution de la chaleur sur une surface, ce qui peut être crucial pour détecter des zones de stress thermique ou des points chauds qui pourraient indiquer une faiblesse ou une dégradation matérielle.
- Cette analyse aide à diagnostiquer des problèmes liés à la fatigue thermique ou à la déformation des matériaux.

4. Fiabilité et Précision :

- Les caméras thermiques offrent une grande précision dans la détection des variations de température, ce qui permet une évaluation fiable de l'état des matériaux.
- La capacité à fournir des données quantitatives sur les températures aide à une analyse plus détaillée et précise des fissures et des défauts.

5. Sécurité et Accessibilité :

- Utiliser un robot équipé d'une caméra thermique pour inspecter des structures à grande hauteur ou dans des environnements dangereux améliore la sécurité des opérateurs en éliminant le besoin de se trouver physiquement dans des zones à risque.
- Les robots grimpeurs peuvent accéder à des endroits difficiles d'accès ou dangereux pour les humains, comme les gratte-ciel, les ponts ou les installations industrielles.

6. Applications Variées :

- Les caméras thermiques sont utilisées dans divers secteurs tels que la construction, la maintenance industrielle, l'énergie, et la sécurité. Leur capacité à détecter des anomalies thermiques les rend polyvalentes pour de nombreuses applications de surveillance et d'inspection.
- Dans le cadre de notre projet, cela permet d'élargir le champ d'application du robot à d'autres types d'inspections, augmentant ainsi sa valeur et son utilité.

➤ En intégrant une <<caméra thermique>> dans notre robot grimpeur, nous dotons notre système d'une capacité avancée de détection et d'analyse des fissures et autres défauts structurels. Cela améliore non seulement l'efficacité et la précision de l'inspection, mais aussi la sécurité et la polyvalence du robot. Ces avantages font de la caméra thermique un choix idéal pour le projet de Robot Grimpeur Détecteur de

Fissures, nous permettant d'identifier et de diagnostiquer des problèmes invisibles autrement, dans des conditions variées et souvent difficiles.

Partie Ilef :

1) Carte Électronique :

Une carte électronique est un support physique sur lequel sont montés des composants électroniques, tels que des circuits intégrés, des transistors, des résistances et des condensateurs, qui sont reliés entre eux par des fils conducteurs. Les cartes électroniques sont utilisées pour intégrer et connecter ces composants de manière à réaliser une fonction spécifique, telle que le traitement de données ou la commande de moteurs.

Caractéristique	Raspberry Pi 4	Raspberry Pi 3	Arduino Uno	BeagleBone Black	ESP32	
Processeur	Quad-core ARM Cortex-A72 à 1.5 GHz	Quad-core ARM Cortex-A53 à 1.2 GHz	ATmega328 P à 16 MHz	ARM Cortex-A8 à 1 GHz	Dual-core Xtensa LX6 à 240 MHz	
Mémoire RAM	2GB, 4GB, ou 8GB LPDDR4	1GB LPDDR2	2 KB SRAM	512 MB DDR3	520 KB SRAM, 448 KB ROM	
Stockage	Carte microSD	Carte microSD	Mémoire flash de 32 KB	4 GB eMMC	Mémoire flash externe (jusqu'à 16 MB)	
Ports USB	2x USB 3.0, 2x USB 2.0	4x USB 2.0	1x USB 2.0	1x USB 2.0, 1x USB OTG	1x USB OTG	

Connectivité réseau	Ethernet Gigabit, Wi-Fi 802.11ac, Bluetooth 5.0	Ethernet 10/100, Wi-Fi 802.11n, Bluetooth 4.1	Aucun	Ethernet 10/100, Wi-Fi via dongle	Wi-Fi 802.11 b/g/n, Bluetooth 4.2	
GPIO	40 broches	40 broches	14 broches numériques, 6 analogiques	69 broches	34 broches	
Vidéo	2x Micro HDMI, support 4K	HDMI, support 1080p	Aucun	Micro HDMI	Aucun	
Système d'exploitation	Linux (Raspbian, Ubuntu, etc.)	Linux (Raspbian, Ubuntu, etc.)	Aucun (programmé avec des sketches)	Linux (Debian, Ubuntu)	MicroPython, FreeRTOS, Arduino IDE	
Prix	Environ 35-75 USD selon la RAM	Environ 35 USD	Environ 20 USD	Environ 60 USD	Environ 5-10 USD	

La Raspberry Pi 4 offre une puissance de traitement supérieure par rapport à la Raspberry Pi 3 et aux autres cartes. Elle dispose d'une mémoire RAM extensible jusqu'à 8GB, bien plus que la Raspberry Pi 3, l'Arduino Uno, le BeagleBone Black et l'ESP32. Sa connectivité réseau avancée inclut Ethernet Gigabit et Wi-Fi 802.11ac, surpassant ainsi la Raspberry Pi 3 et les autres cartes. De plus, la Raspberry Pi 4 supporte la vidéo 4K avec deux sorties Micro HDMI, une caractéristique unique parmi les cartes listées.

Références :

- <https://www.quora.com/Which-is-better-for-robotics-research-Arduino-Raspberry-Pi-or-BeagleBone-Black>
- <https://www.raspberrypi.com/documentation/computers/raspberry-pi.html>
- <https://www.beagleboard.org/boards/beaglebone-black>
- <https://www.espressif.com/en/products/socs/esp32/resources>

2) Langage de programmation :

Caractéristique	Python	C#
Facilité d'apprentissage	Très facile	Moyenne
Rapidité de développement	Très rapide	Moyenne
Gestion de la mémoire	Automatique (gestion par garbage collector)	Automatique (gestion par garbage collector)
Bibliothèques disponibles	Très nombreuses	Nombreuses
Support pour les projets robotiques	Excellent	Bon
Communauté et documentation	Très large	Large
Interopérabilité	Très élevée (interopérabilité avec C/C++)	Élevée
Portabilité	Très élevée	Moyenne
Paradigmes de programmation	Multi-paradigme (procédural, OO, fonctionnel)	Orienté objet
Utilisation en éducation	Très courant	Courant

=> Python est particulièrement adapté pour les projets robotiques en raison de sa facilité d'apprentissage, de sa rapidité de développement, de la richesse de ses bibliothèques spécialisées, de sa portabilité et de son large support communautaire. Ces avantages en font un choix optimal par rapport à C# pour des projets nécessitant une flexibilité et une efficacité de développement accrues.

Références :

- <https://realpython.com/micropython/>
- <https://www.geeksforgeeks.org/top-programming-languages-for-competitive-programming/>

3) Caméra thermique :

Le principe de la caméra thermique

Le principe d'une caméra thermique est de mesurer et d'enregistrer les différentes ondes de chaleur, rayonnements infrarouges, émis par un corps ou un objet. Elle reproduit une image représentant l'intensité du rayonnement, ce qui permet d'évaluer la température. En effet, plus la température d'un corps ou d'un objet est élevée, plus le rayonnement est important. Ce sont ces données qui permettent aux caméras thermiques de restituer une cartographie spatiale des températures, appelée thermographe



Caméras thermiques - Source Testo

Bâtiment

- détection des points faibles de l'isolation thermique d'un bâtiment ;
- vérification des températures des canalisations et installations de chauffage, notamment pour le contrôle des planchers chauffants ;
- vérification des armoires électriques par visualisation des surchauffes des connexions, ou de certains composants.

Critère	Caméra Thermique	Caméra Visible
Capacité de détection des fissures	Très élevée (détection de différences de température, y compris les fissures internes)	Moyenne (dépend des conditions de lumière et de l'angle)
Détection des défauts internes	Oui (détection des variations de chaleur causées par des fissures internes)	Non
Indépendance des conditions lumineuses	Oui (fonctionne dans l'obscurité totale)	Non (dépend fortement des conditions lumineuses)
Portée de détection	Moyenne (quelques mètres)	Élevée (dépend de la résolution)

		de la caméra)
Précision	Élevée (dépend de la résolution thermique)	Moyenne (dépend de la qualité de l'image et de la lumière)
Coût	Moyen à élevé	Faible à moyen
Facilité d'intégration	Moyenne (nécessite un traitement d'image spécifique)	Facile
Utilisation en industrie	Courante (inspection thermique des bâtiments, maintenance préventive)	Courante (inspection visuelle)

- En conclusion, dans notre projet de robot grimpeur détecteur de fissures, la comparaison entre une caméra visible et une caméra thermique met en évidence les avantages significatifs de cette dernière. La caméra thermique offre une détection plus précise des fissures, y compris celles invisibles à l'œil nu, grâce à sa capacité à détecter les variations de température. Sa capacité à fonctionner indépendamment des conditions lumineuses la rend également plus polyvalente et fiable dans divers environnements. Ainsi, l'utilisation d'une caméra thermique dans notre projet s'avère plus bénéfique pour assurer une détection efficace et précoce des fissures, contribuant ainsi à la maintenance proactive des structures.

Références :

- <https://www.xpair.com/lexique/definition/camera-thermique.htm>
- <https://www.flir.quebec/discover/ots/thermal-vs-night-vision/#:~:text=La%20nuit%2C%20lorsque%20l'absence,r%C3%A9fl%C3%A9chie%20%3A%20elles%20voient%20la%20chaleur.>

PARTIE ZEINEB/

Pourquoi Linux plutôt que Windows

1. Performance et stabilité :

- Linux est connu pour sa stabilité et sa capacité à gérer des tâches de manière plus efficace que Windows, en particulier sur des systèmes embarqués comme Raspberry Pi et Jetson Nano.

- Moins de ressources système sont utilisées pour l'interface graphique sous Linux, permettant plus de ressources disponibles pour les applications critiques comme le traitement d'images.

2. Open Source :

- Linux est open source, ce qui signifie que vous pouvez personnaliser et modifier le système d'exploitation selon vos besoins spécifiques.

- Large accès à une multitude de logiciels libres et de bibliothèques nécessaires pour la robotique et le traitement d'images.

3. Support de la communauté et documentation :

- Une grande communauté de développeurs et d'utilisateurs de Linux peut offrir une aide précieuse et des solutions à des problèmes techniques.

- Documentation abondante et forums de support pour les matériels et logiciels utilisés dans la robotique.

4. Compatibilité et Flexibilité :

- La majorité des logiciels et bibliothèques nécessaires pour le traitement d'images, la vision par ordinateur et l'intelligence artificielle (OpenCV, Tensor Flow, etc.) sont mieux supportés et souvent d'abord développés pour Linux.

- Facilite l'intégration avec des outils de développement et de déploiement couramment utilisés dans l'industrie (Docker, Kubernetes).

5. Sécurité :

- Linux est souvent considéré comme plus sécurisé que Windows. Les mises à jour de sécurité sont rapides et moins susceptibles de nécessiter des redémarrages fréquents.

Comparaison Linux vs Windows

Caractéristique	Linux	Windows
Performance	Plus efficace, moins ressources utilisées	Beaucoup plus de ressources
Open source	oui	non
support de la communauté	Excellent	Moins orienté open source
Stabilité	très stable	bon mais peut nécessiter des redémarrages fréquents
Compatibilité Logicielle	Large support pour les logiciels embarqués	moins robotique ou embarqué

sécurité	très bonne	Bonne, mais les mise à jour peuvent être disruptives
----------	------------	--

cross compilation

La cross-compilation est le processus de compilation de code source sur une plate-forme (le système hôte:linux) pour qu'il soit exécuté sur une autre plate-forme différente (le système cible:windows). Cela est particulièrement utile lorsque le système cible a des ressources limitées ou une architecture différente de celle du système hôte.

Différence d'architecture :

Les systèmes embarqués, comme ceux utilisant des cartes Raspberry Pi ou NVIDIA Jetson, ont souvent des architectures de processeur différentes de celles des ordinateurs de bureau. La cross-compilation permet de compiler le code sur une architecture (par exemple, x86) pour qu'il fonctionne sur une autre (par exemple, ARM).

Ressources limitées :

Les systèmes embarqués ont généralement des ressources limitées en termes de puissance de traitement, de mémoire et de stockage. Compiler directement sur ces systèmes peut être très lent et inefficace. La cross-compilation permet d'utiliser les ressources plus puissantes du système hôte pour effectuer la compilation.



04/06/2024

Quel est le meilleur langage de programmation à apprendre pour faire de la robotique ?

Je dirai qu'il en faut au moins deux, C/C++ pour le codage bas niveau sur microcontrôleurs et python ou javascript pour le codage haut niveau de la partie mission, IA communications etc... car ce sont des langage très répandus dans les communautés robotiques, avec de nombreuses bibliothèques très rapides .Dans la partie haut niveau on se fiche de la rapidité du langage car on passe très peu de temps dedans, c'est les bibliothèques qui comptent.

Par contre c'est très important à haut niveau pour des robots versatiles d'avoir des langages à objets dynamiques et acceptant du code à la volée.

Vous pouvez programmer un aspirateur robot ou un chasseur de mine entièrement dans un langage objet figé avec des classes, mais un robot plus versatile aura besoin de code dynamique.

Après selon le type de robotique il faut maîtriser pas mal du côté, des langages de codage de données.comme json ou XML, le grafcet pour les robots industriels etc

Avantages pour python :

<https://niryo.com/fr/learn-robotics/5-benefits-of-using-python-in-robotics/#:~:text=M%C3%A4me%20s'il%20n'existe,haut%20niveau%20les%20plus%20populaires.>

Partie chaima :

1) Carte Raspberry Pi 4 :

La carte Raspberry Pi 4 offre une fréquence de processeur plus élevée et une gestion optimisée de l'alimentation :

1. Fréquence de processeur :

- La Raspberry Pi 4 est équipée d'un processeur quad-core Cortex-A72 (ARM v8) 64-bit, cadencé à 1,5 GHz.

2. Alimentation :

- Bien que la Raspberry Pi 4 soit plus puissante, elle nécessite une alimentation légèrement plus élevée pour fonctionner de manière optimale. Elle utilise un port USB-C pour l'alimentation, avec une recommandation d'utiliser une alimentation de 5V/3A. Cette augmentation de l'exigence en alimentation

permet de prendre en charge le matériel plus puissant et les fonctionnalités supplémentaires sans compromettre la stabilité du système.

- La Raspberry Pi 4 offre effectivement une fréquence de processeur augmentée et, bien qu'elle nécessite une alimentation légèrement plus puissante par rapport aux modèles précédents, cette alimentation reste relativement faible en comparaison avec des ordinateurs traditionnels.

2) Langage de programmation :

Pour notre projet de robot grimpeur détecteur de fissures utilisant une carte Raspberry Pi 4, des bras robotisés et une caméra thermique, il est essentiel de choisir un langage de programmation (ou plusieurs) qui offre des bibliothèques robustes, une bonne communauté de support et la capacité d'interagir efficacement avec le matériel.

❖ **Python**

Avantages

1. **Bibliothèques Abondantes** : Python a une vaste collection de bibliothèques pour l'interaction avec le matériel, comme **RPi.GPIO** pour contrôler les GPIO de la Raspberry Pi, **opencv** pour le traitement d'images (y compris les caméras thermiques), et **pyserial** pour la communication série.
2. **Facilité d'Utilisation** : Python est réputé pour sa syntaxe simple et claire, ce qui facilite le développement rapide et le prototypage.
3. **Communauté et Documentation** : Une grande communauté de développeurs Python signifie qu'il existe une abondance de tutoriels, de forums d'aide, et de projets similaires sur lesquels vous pouvez vous baser.

Inconvénients

- **Performance** : Python peut être plus lent que les langages compilés comme C ou C++, mais pour la majorité des applications sur Raspberry Pi, la différence de performance n'est pas significative.

❖ **C/C++**

Avantages

1. **Performance** : C et C++ sont des langages compilés, donc ils offrent de meilleures performances et une utilisation plus efficace des ressources matérielles.
2. **Contrôle Bas-Niveau** : Ces langages offrent un contrôle plus fin sur le matériel, ce qui peut être crucial pour les applications nécessitant une manipulation précise des GPIO et des timings critiques.
3. **Bibliothèques Robustes** : Il existe de nombreuses bibliothèques pour l'interaction avec le matériel, comme **wiringPi** et **pigpio** pour les GPIO, ainsi que OpenCV pour le traitement d'images en C++.

Inconvénients

- **Complexité** : C et C++ ont une courbe d'apprentissage plus abrupte et nécessitent une gestion explicite de la mémoire, ce qui peut augmenter la complexité du développement.

➤ Utilisation de Plusieurs Langages

Il est tout à fait possible d'utiliser plusieurs langages dans un même projet pour tirer parti des avantages spécifiques de chaque langage :

1. **Python pour le Prototypage et les Composants de Haut Niveau** :
 - Utilisez Python pour les tâches de haut niveau comme le traitement d'images et la logique générale du robot. La facilité de développement et les nombreuses bibliothèques disponibles rendent Python idéal pour ces tâches.
2. **C/C++ pour les Composants de Bas Niveau et Critiques en Temps Réel** :
 - Utilisez C ou C++ pour les composants critiques en temps réel où la performance est essentielle, comme le contrôle des bras robotisés et la gestion des capteurs nécessitant des timings précis.

➤ Communication entre les Langages

Vous pouvez utiliser plusieurs méthodes pour permettre la communication entre les différents composants développés en différents langages :

- **API RESTful** : Développez des services en Python qui exposent des API RESTful, et appelez ces services depuis des programmes en C/C++.
- **Interfaces C/Python (Cython, ctypes, cffi)** : Utilisez des outils comme Cython ou ctypes pour appeler du code C/C++ depuis Python.
- **Message Passing** : Utilisez des files de messages ou des sockets pour permettre la communication entre des programmes Python et C/C++.

➤ Conclusion pour langage de programmation

Pour notre projet de robot grimpeur détecteur de fissures, une approche hybride utilisant Python et C/C++ serait probablement la plus efficace. Python vous permettra de développer rapidement et d'utiliser de nombreuses bibliothèques disponibles, tandis que C/C++ vous donnera le contrôle et la performance nécessaires pour les composants critiques.

1. **Python pour** :
 - Le traitement d'images et l'analyse des données de la caméra thermique (avec OpenCV).
 - La logique globale de contrôle du robot.
2. **C/C++ pour** :
 - Le contrôle précis des bras robotisés.

- La gestion des capteurs et des actionneurs nécessitant des timings critiques.
- Cette combinaison vous permettra de bénéficier du meilleur des deux mondes et de créer un système performant et flexible.

Les Bibliothèques disponibles en Python :

Pour programmer un robot en utilisant le langage Python, plusieurs bibliothèques et frameworks sont disponibles, chacune adaptée à des besoins spécifiques en termes de complexité, d'environnement et de type de robot. Voici une liste des bibliothèques les plus couramment utilisées :

1. Robot Operating System (ROS)

- **Description:** ROS est une plateforme flexible pour l'écriture de logiciels de robotique. Il inclut des outils et des bibliothèques pour construire des systèmes robotiques complexes.
- **Bibliothèque Python:** `rospy`
- **Site Web:** ros.org

2. PyRobot

- **Description:** Une interface de haut niveau pour ROS, développée par Facebook AI Research. Elle facilite le contrôle de divers robots sans nécessiter de connaissances approfondies en ROS.
- **Site Web:** pyrobot.org

3. Pygame

- **Description:** Utilisé pour créer des simulations robotiques 2D simples et pour le prototypage rapide.
- **Site Web:** pygame.org

4. VPL (Visual Programming Language) pour robots LEGO Mindstorms

- **Description:** Utilisé pour programmer les robots LEGO Mindstorms en Python.
- **Bibliothèque Python:** `ev3dev`
- **Site Web:** ev3dev.org

5. pybullet

- **Description:** Un module Python pour la simulation de physique en temps réel (Rigidbody Dynamics, Contrainte, Collision Detection). Utilisé pour la simulation de robots.
- **Site Web:** pybullet.org

6. OpenCV

- **Description:** Utilisé pour la vision par ordinateur dans les projets robotiques. Il permet de traiter des images et des vidéos en temps réel.
- **Site Web:** opencv.org

7. TensorFlow / PyTorch

- **Description:** Utilisés pour l'apprentissage automatique et la vision par ordinateur avancée. Ces bibliothèques peuvent être intégrées dans des projets robotiques pour la reconnaissance d'objets, la navigation, etc.
- **Sites Web:** [tensorflow.org](https://www.tensorflow.org), pytorch.org

8. Scipy / Numpy

- **Description:** Bibliothèques pour le calcul scientifique et numérique. Très utiles pour la modélisation mathématique et les calculs nécessaires dans la robotique.
- **Sites Web:** [scipy.org](https://www.scipy.org), numpy.org

9. Raspberry Pi GPIO

- **Description:** Utilisé pour interfacer un Raspberry Pi avec des capteurs et des actionneurs externes.
- **Bibliothèque Python:** `RPi.GPIO`
- **Site Web:** [raspberrypi.org](https://www.raspberrypi.org)

Ces bibliothèques offrent un large éventail de fonctionnalités pour répondre à divers besoins en robotique, de la simulation et la planification de mouvements à l'interfaçage matériel et l'intelligence artificielle.

Les Bibliothèques disponibles en C :

Bibliothèques pour la Gestion des GPIO

wiringPi

- Une bibliothèque populaire pour la gestion des GPIO sur Raspberry Pi.
- **Documentation**
: <https://www.framboise314.fr/la-documentation-de-wiringpi-traduite-en-francais/#:~:text=WiringPi%20est%20l'une%20des,des%20p%C3%A9riph%C3%A9riques%20les%20plus%20populaires>.

pigpio

- Une bibliothèque pour le contrôle des GPIO avec une granularité temporelle fine.
- **Documentation :** <https://abyz.me.uk/rpi/pigpio/>

Bibliothèques pour la Communication

libserial

- Une bibliothèque C pour la communication série.
- **Documentation :**
<https://libserial.readthedocs.io/en/latest/description.html>

libi2c-dev

- Une bibliothèque pour la communication I2C, disponible dans les dépôts de paquets de Raspbian.
- Documentation : <https://packages.debian.org/buster/libi2c-dev>

Bibliothèques pour les Bras Robotisés

ROS (Robot Operating System)

- Bien que ROS soit un cadre logiciel plus qu'une simple bibliothèque, il fournit des outils et des bibliothèques pour la conception de robots. Il est compatible avec C++ et Python.
- Documentation : <https://ros.org/>

Dynamixel SDK

- Une bibliothèque pour contrôler les moteurs Dynamixel, souvent utilisés dans les bras robotisés.
- Documentation : https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_sdk/overview/

Exemple d'Utilisation

1. **Contrôle des Bras Robotisés :**
 - Utilisez **wiringPi** ou **pigpio** pour les GPIO.
 - Si vous utilisez des moteurs spécifiques comme les Dynamixel, utilisez le Dynamixel SDK.
2. **Acquisition et Traitement d'Images Thermiques :**
 - Utilisez le Lepton SDK pour interfacier avec la caméra thermique.
 - Utilisez OpenCV pour le traitement d'images et la détection des fissures.
3. **Gestion des Capteurs et des Données :**
 - Utilisez les bibliothèques Adafruit pour interfacier avec d'autres capteurs nécessaires.
 - Utilisez **libserial** ou **libi2c-dev** pour la communication avec d'autres composants électroniques via les protocoles série ou I2C.

Partie Zeineb:

Les Bibliothèques disponibles en C++

Robot Operating System (ROS)

Description: ROS est une plateforme flexible pour l'écriture de logiciels de robotique. Il inclut des outils et des bibliothèques pour construire des systèmes robotiques complexes.

Bibliothèque C++: roscpp

Site Web: ros.org

MoveIt

Description: MoveIt est une bibliothèque pour la planification de mouvements, largement utilisée avec ROS pour contrôler les bras robotiques.

Site Web: moveit.ros.org

Gazebo

Description: Gazebo est un simulateur de robot open-source avec une interface C++ pour la simulation de physique réaliste, l'interaction de capteurs et la visualisation.

Site Web: gazebo.org

PCL (Point Cloud Library)

*Description: PCL est une bibliothèque pour le traitement des nuages de points 3D, souvent utilisée en robotique pour la perception.

Site Web: pointclouds.org

OpenCV

*Description: Utilisé pour la vision par ordinateur dans les projets robotiques. Il permet de traiter des images et des vidéos en temps réel.

Site Web: opencv.org

Dlib

*Description: Dlib est une boîte à outils pour l'apprentissage automatique et la vision par ordinateur, utile pour l'intégration de modèles de machine learning dans les projets robotiques.

Site Web: dlib.net

Bullet Physics SDK

*Description: Bullet est une bibliothèque pour la simulation de physique en temps réel (Rigidbody Dynamics, Contrainte, Collision Detection). Utilisé pour la simulation de robots.
Site Web: bulletphysics.org

Eigen

*Description: Eigen est une bibliothèque de calcul matriciel et algébrique pour C++. Elle est très utilisée pour les calculs mathématiques en robotique.
Site Web: eigen.tuxfamily.org

TensorFlow / PyTorch C++ API

*Description: TensorFlow et PyTorch offrent des API C++ pour l'apprentissage automatique et la vision par ordinateur avancée. Ces bibliothèques peuvent être intégrées dans des projets robotiques pour la reconnaissance d'objets, la navigation, etc.
Sites Web: tensorflow.org, pytorch.org

WiringPi

*Description: WiringPi est une bibliothèque pour interfacer un Raspberry Pi avec des capteurs et des actionneurs externes.
Site Web: wiringpi.com

Boost

*Description: Boost fournit des bibliothèques gratuites peer-reviewed portables C++ pour le développement d'applications robotiques, y compris les opérations mathématiques, les structures de données, les algorithmes et plus encore.
Site Web: boost.org

1. Kinova Robotics JACO Arm

- *Description: Le bras JACO est conçu pour être léger, compact et adaptable à diverses applications, y compris celles nécessitant une haute précision et une manipulation délicate.*
- *Caractéristiques:*
 - *Charge utile : 2 à 5 kg*
 - *Portée : Jusqu'à 900 mm*
 - *Programmation : API disponible pour Python et ROS*
- Site Web: kinovarobotics.com

2. DOBOT Magician

- *Description: Un bras robotique de bureau polyvalent et abordable, adapté pour des projets éducatifs, de recherche et des applications légères.*
 - *Caractéristiques:*
 - *Charge utile : 0.5 kg*
 - *Portée : 320 mm*
 - *Programmation : Supporte Python, Blockly, et ROS*
 - *Site Web: dobot.cc*
- 3. Franka Emika Panda**
- *Description: Un bras robotique léger et précis, conçu pour être facile à utiliser et à intégrer dans diverses applications industrielles et de recherche.*
 - *Caractéristiques:*
 - *Charge utile : 3 kg*
 - *Portée : 850 mm*
 - *Programmation : API disponible pour Python, ROS*
 - *Site Web: franka.de*
- 4. Trossen Robotics Interbotix Arms**
- *Description: Trossen Robotics propose une gamme de bras robotiques, notamment les Interbotix Arms, adaptés pour la recherche, l'éducation et les applications légères.*
 - *Caractéristiques:*
 - *Charge utile : Varie selon les modèles (0.75 kg à 1.5 kg)*
 - *Portée : Varie selon les modèles (300 mm à 500 mm)*
 - *Programmation : Supporte Python, ROS*
 - *Site Web: trossenrobotics.com*

Ces bras robotiques peuvent être intégrés avec des systèmes de vision pour la détection de fissures, utilisant des caméras haute résolution et des algorithmes de traitement d'image comme ceux disponibles dans OpenCV. En fonction des spécificités du projet (environnement de travail, taille et nature des fissures, mobilité nécessaire), le choix du bras et de l'équipement associé pourra être affiné.

Le 18 DOF-Robot Araignée Hexapode est un robot à six pattes, conçu pour des projets de bricolage, de jouets ou de projets éducatifs. "18 DOF" signifie "18 degrés de liberté", ce qui indique qu'il a 18 axes de mouvement, offrant une grande flexibilité et des mouvements complexes.

Caractéristiques

1. *Nombre de pattes : Six pattes (hexapode), ce qui lui permet une grande stabilité et la capacité de se déplacer sur divers terrains.*
2. *Degrés de liberté (DOF) : Chaque patte dispose de trois degrés de liberté, permettant des mouvements fluides et réalistes.*
3. *Matériau : Généralement fait de matériaux légers mais robustes pour un bon équilibre entre durabilité et performance.*
4. *Servo-moteurs : Équipé de servos pour contrôler les mouvements des pattes. Ces servos sont souvent commandés par une carte Arduino ou une autre plateforme de microcontrôleur.*
5. *Contrôle : Peut être contrôlé via une télécommande RC (Radio Control) ou programmé pour des mouvements autonomes.*
6. *Poids : Environ 379 grammes, ce qui le rend relativement léger et facile à manipuler.*
7. *Applications : Utilisé pour l'apprentissage de la robotique, des projets de bricolage, des démonstrations scientifiques, et comme jouet éducatif.*

Intégrer une carte raspberry

Oui, il est tout à fait possible d'intégrer une carte Raspberry Pi 4 à un robot hexapode à 18 DOF. La Raspberry Pi 4 peut fournir des capacités de traitement supplémentaires, permettant des applications plus avancées telles que la vision par ordinateur, l'intelligence artificielle, et des algorithmes de contrôle sophistiqués. Voici comment vous pouvez procéder :

Matériel Nécessaire

1. *Raspberry Pi 4 : La carte elle-même.*

2. *Alimentation : Une source d'alimentation fiable pour la Raspberry Pi 4, généralement une alimentation USB-C.*
3. *Carte MicroSD : Pour installer le système d'exploitation de la Raspberry Pi.*
4. *Servo HAT ou contrôleur PWM : Pour contrôler les servo-moteurs du robot hexapode. Un contrôleur PWM comme le PCA9685 est souvent utilisé.*
5. *Câbles de connexion : Pour connecter les servos au contrôleur PWM et pour interfacer le contrôleur avec la Raspberry Pi.*
6. *Batterie : Pour alimenter le robot et les servos. Une batterie LiPo est souvent utilisée.*

Étapes d'Intégration

1. *Préparation de la Raspberry Pi :*
 - *Installez le système d'exploitation Raspberry Pi OS sur une carte MicroSD et configurez-le.*
 - *Assurez-vous que la Raspberry Pi a une connexion réseau pour pouvoir installer des logiciels supplémentaires.*
2. *Installation du contrôleur PWM :*
 - *Connectez le contrôleur PWM (comme le PCA9685) aux broches GPIO de la Raspberry Pi. Suivez le schéma de câblage du fabricant pour les connexions.*
 - *Connectez les servos du robot hexapode au contrôleur PWM.*
3. *Alimentation :*
 - *Assurez-vous que la Raspberry Pi et les servos reçoivent une alimentation adéquate. Utilisez une batterie LiPo pour les servos et une alimentation USB-C pour la Raspberry Pi.*
 - *Vérifiez les tensions et les courants pour éviter toute surcharge.*
4. *Programmation :*
 - *Installez les bibliothèques nécessaires pour contrôler les servos. Par exemple, vous pouvez utiliser la bibliothèque `Adafruit_PCA9685` pour Python.*
 - *Écrivez un programme pour contrôler les mouvements des servos en utilisant la Raspberry Pi. Vous pouvez commencer par des mouvements simples et graduellement ajouter des comportements plus complexes.*
 - *Si vous souhaitez ajouter des capacités de vision par ordinateur, vous pouvez installer OpenCV et développer des algorithmes pour la détection et la reconnaissance d'objets.*
5. *Test et Débogage :*
 - *Testez chaque servo individuellement pour vous assurer qu'il fonctionne correctement.*

- *Testez les mouvements de base du robot pour vérifier la synchronisation et la coordination des servos.*
- *Déboguez et ajustez votre programme en fonction des résultats des tests*

L'architecture ARM (Advanced RISC Machine) est une famille d'architectures de processeur basées sur le modèle RISC (Reduced Instruction Set Computing). Les processeurs ARM sont largement utilisés dans une variété d'applications, notamment dans les appareils mobiles, les systèmes embarqués, et plus récemment dans les serveurs et les ordinateurs personnels.

Principes de Base de l'Architecture ARM

1. RISC (Reduced Instruction Set Computing) :

- L'architecture ARM utilise un ensemble réduit d'instructions par rapport aux architectures CISC (Complex Instruction Set Computing) comme x86. Cela permet une exécution plus rapide des instructions grâce à une conception plus simple et à une consommation d'énergie réduite.

2. Performance et Efficacité Énergétique :

- Les processeurs ARM sont réputés pour leur efficacité énergétique, ce qui les rend idéaux pour les appareils portables où la durée de vie de la batterie est cruciale.

3. Architecture Modulaire :

- ARM propose une architecture modulaire qui permet aux fabricants de processeurs de personnaliser et de configurer des cœurs de processeur en fonction des besoins spécifiques des applications.

Caractéristiques Techniques

1. Registres :

- ARM utilise un ensemble de registres large et uniforme, avec un accent particulier sur les registres généraux. La plupart des instructions peuvent utiliser n'importe quel registre, ce qui simplifie le compilateur et améliore les performances.

2. Instruction Set :

- L'ISA (Instruction Set Architecture) d'ARM comprend des instructions simples et fixes de 32 bits (et parfois 16 bits avec Thumb) qui sont faciles à décoder et à exécuter.

3. Pipeline :

- Les processeurs ARM utilisent un pipeline pour améliorer les performances en exécutant plusieurs instructions simultanément à différentes étapes de leur traitement.

4. Modes de Fonctionnement :

- ARM dispose de plusieurs modes de fonctionnement (User, FIQ, IRQ, Supervisor, Abort, Undefined, et System) pour gérer les interruptions, les exceptions et les privilèges du système.

Utilisations Courantes

1. **Appareils Mobiles :**

- ARM est dominant dans le secteur des smartphones et des tablettes grâce à son efficacité énergétique et à sa puissance de calcul suffisante pour ces applications.

2. **Systèmes Embarqués :**

- Les processeurs ARM sont largement utilisés dans les systèmes embarqués, tels que les microcontrôleurs, les appareils IoT, les téléviseurs, et les appareils électroménagers intelligents.

3. **Ordinateurs Personnels et Serveurs :**

- Avec des avancées comme les processeurs Apple M1 et M2, ARM gagne du terrain dans les ordinateurs personnels et les serveurs, offrant des performances compétitives avec une consommation d'énergie réduite.

Évolution et Innovations

1. **ARMv8 et ARM9 :**

- ARMv8 a introduit le support 64 bits, augmentant les capacités de traitement et la mémoire adressable. ARMv9 continue d'améliorer la sécurité, l'intelligence artificielle, et les performances générales.

2. **Big.LITTLE :**

- La technologie big.LITTLE d'ARM combine des cœurs de processeur à haute performance et à faible consommation d'énergie pour optimiser la performance et l'efficacité énergétique selon les besoins des tâches.

Écosystème et Licences

1. **Licences :**

- ARM Holdings, la société derrière ARM, ne fabrique pas elle-même de processeurs mais vend des licences de son architecture à d'autres entreprises qui conçoivent et fabriquent leurs propres puces basées sur ARM.

2. **Écosystème :**

- ARM dispose d'un large écosystème de partenaires et de développeurs, avec un soutien étendu pour les outils de développement, les systèmes d'exploitation (comme Android, Linux, et Windows), et une communauté active de contributeurs.

Conclusion

L'architecture ARM est extrêmement polyvalente et est supportée par une grande variété de systèmes d'exploitation couvrant de nombreux domaines d'application. Cette flexibilité contribue à l'adoption généralisée de l'architecture ARM dans des

secteurs allant des appareils mobiles aux serveurs en passant par les systèmes embarqués et l'IoT.