# Welke programmeertaal is het meest geschikt om de robotarm te programmeren?

(Deelvraag 5)

Het kiezen van de juiste programmeertaal voor het project is een belangrijke (onderzoeks) stap. De taal op zich is het fundament van je applicatie. Wanneer de verkeerde taal gebruikt wordt, kan achteraf blijken dat het systeem niet of slecht zal werken. Ook kan het gebruik van de juiste programmeertaal veel tijd besparen.

## Real-time

Bij Technische Informatica is real-time een belangrijk begrip. Het principe real-time (in de scope van het project ) betekent dat de robot gegarandeerd reageert binnen een gestelde tijd.

### Besturingssysteem

Wanneer er gebruik wordt gemaakt van een (normaal) besturingssysteem kan het zijn dat je programma door de scheduler of achtergrondprocessen vertraagd of onderbroken wordt.

Een real-time operating system (verder RTOS genoemd) kan zorgen voor een gegarandeerde uitvoertijd van een programma.

### Programmeertaal

In de technische achtergrond van een programmeertaal kunnen ook elementen zitten die ervoor kunnen zorgen dat een programmeertaal wel of niet real-time is. De Java garbage collector kan er bijvoorbeeld voor zorgen dat het programma vertraging oploopt.

## Communicatie

Communiceren met de robot gaat via een USB -> RS232 connector. Wanneer deze wordt aangesloten (en de juiste drivers worden geïnstalleerd) zal de PC dit identificeren als een COM poort.

Het is dus belangrijk dat de programmeertaal communicatie via een COM poort ondersteunt.

## Objecttracking

Voor het project is het nodig om objecten te kunnen traceren door middel van video-interpretatie. Het is belangrijk dat de taal hiervoor een mogelijkheid ondersteunt. Ook is het wenselijk dat het traceren van een object ‘snel’ gaat.

## User base/support

Het is erg wenselijk dat de programmeertaal een grote gebruikersbasis heeft en mogelijkheid tot support of help van een community.

## De programmeertaal

Met oog op de verscheidenen eisen kunnen programmeertalen met elkaar vergeleken. Hiermee kan worden bepaald welke programmeertaal geschikt is voor onze opdracht.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Taal | Real-time (hard) | Operating systems | Communicatie | Object tracking | Userbase (0 tot 5) |
| C | ✓ | Indien compiler beschikbaar | ✓ | ✓ | 4 |
| C++ | ✓ | Indien compiler beschikbaar | ✓ | ✓ | 3 |
| Java | x | Draait enkel op JVM | ✓ | ✓ | 5 |
| Real Time Java | ✓ | Een aantal geselecteerde | ✓ | x | 1 |
| C# | x | Draait op .NET | ✓ | ✓ | 4 |
| Python | x | Een aantal geselecteerde | ✓ | ✓ | 3 |

### Toelichting bij tabel:

**Real-time:**

Bij real-time wordt onderscheid gemaakt tussen “hard” en “soft” real-time. Bij hard real-time is het halen van een deadline cruciaal. Mocht de deadline niet gehaald worden dan zal dit leiden tot schade. Bij soft real-time is het wenselijk dat het systeem bijvoorbeeld 99% van de tijd op tijd is. Daarbij is het 1x missen van een deadline niet gevaarlijk maar vaak wel jammer. Bij soft real-time kan een Windows OS voldoende moeten zijn.

C/C++ is hard real-time (wanneer uitgevoerd op een RTOS). Dit wil zeggen dat de timing deterministisch/berekenbaar. C/C++ wordt gecompileerd naar machinetaal/assembler die direct uitgevoerd kan worden. Er is geen garbage collector of virtual machine die het programma pauzeren.

Java/C#(.NET)/Python is per definitie niet real-time. De code draait op een virtual machine, deze zorgt voor run-time optimalisatie en garbage collection. Door het run-time optimaliseren/compileren is er niets of weinig te zeggen over hoe de code wordt uitgevoerd en kan daarnaast de garbage collection het programma pauzeren.

**Operating system:**

Voor C/C++ geldt dat elke architectuur waarvoor een compiler beschikbaar is in principe gebruikt kan worden. Bij Java/C# kan elk operating system gebruikt worden mits daar de virtual machine op kan draaien. Bij een RTOS wordt C/C++ meestal ondersteunt aangezien deze ook real-time gedrag kan implementeren. Ook is het in sommige gevallen mogelijk om Java/C#/Python te gebruiken en zal dit de tijdsafwijking verminderen tegenover een niet real-time OS.

Onze voorkeur gaat naar een real-time programmeertaal met deterministische timing (hard real-time). Het reageren van de robot moet voorspelbaar zijn om zeker te zijn dat hij het balletje altijd raakt. Een niet real-time programmeertaal kan op een verantwoorde wijze gebruikt worden, echter zal het invoegen van een library heel waarschijnlijk dit gedrag wegnemen.

Programmeertalen index:  
<http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>

<http://spectrum.ieee.org/computing/software/the-2015-top-ten-programming-languages>

<http://redmonk.com/sogrady/2015/07/01/language-rankings-6-15/>

Real-time operating system on x86 by intel: <http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/multicore-real-time-linux-xenomai-paper.pdf>

Java virtual machine: <https://nl.wikipedia.org/wiki/Java_Virtual_Machine>  
(Normaal) Java/.Net niet hard real-time: <http://stackoverflow.com/questions/3559878/is-a-garbage-collector-net-java-a-problem-for-real-time-systems>

RTOS voor Real Time Java: <http://rtos.com/partners/category/tools#aicas_realtime>

C# native linux: <http://www.itwriting.com/blog/8361-microsofts-new-open-source-direction-for-c-and-net-and-native-compilation-too-anders-hejslberg-explains.html>

# Conclusie

## Programmeertaal

Uiteindelijk is C++ de beste keuze voor het programmeren van een aansturingsprogramma voor de robot. Door C++ te gebruiken kunnen we ook in C schrijven. C++ compilers kunnen in zekere zin ook C compileren. Er zijn kleine (vaak syntactische) verschillen. [http://www.cprogramming.com/tutorial/c-vs-c++.html] Ook kan in C++ inline assembly gebruikt worden [http://www.codeproject.com/Articles/15971/Using-Inline-Assembly-in-C-C].

C++ kan worden gecompileerd naar machine instructies, in principe kan bij C/C++ bepaald worden wat de maximale executietijd van een blok code is (aantal instructies\*tijd per instructie). Daarnaast is het atomisch uitvoeren van code (in theorie) mogelijk. Men weet dan zeker dat twee instructies elkaar volgen. Het vervelende is wel dat een besturingssysteem de processor kan kapen waarbij tussendoor andere code wordt uitgevoerd. Hierdoor zal er dus in de praktijk “jitter” ontstaan (meer hierover bij RTOS). Real-time Java was in ons onderzoek een andere kanshebber. RT-Java is wel een interessante mogelijkheid, maar het is onbekend of de videoherkenning libraries ermee compatibel zijn.

Ook voldoet C++ aan de eisen om COM-poort ondersteuning en videoherkenning te hebben. En ten slot is C++ een bekende taal. Hierdoor zijn er meer mensen te vinden die antwoord kunnen geven op vragen. Als met een onbekende taal gewerkt wordt, nemen we het risico dat niemand weet waarom iets niet werkt.

Keuze voor C++:

* Compatibiliteit met C (en assembly)
* Real-time
* Draait niet op een Virtual Machine (extra laag op OS)
* Benodigde randvoorwaarden behaald (COM-poort ondersteuning en videoherkenning libraries
* Goede user base

## Operating system

Tijdens het onderzoek kwamen we tot de conclusie dat een operating system (vooral Windows) jitter kan introduceren. Het probleem hierbij is dat Windows op de achtergrond taken uitvoert en de performance van het systeem kunnen benadelen.

De mogelijkheid om Linux als OS te gebruiken is ook onderzocht. Op een internet-artikel is beschreven hoe je Linux afstemt voor (soft) real-time gedrag (<https://rt.wiki.kernel.org/index.php/HOWTO:_Build_an_RT-application>). Ook wordt in deze bron beschreven dat de x86 structuur al interne service management interrupts (SMI) heeft. Deze kunnen niet gezien of bewerkt worden door het OS.

Linux configureren voor real-time operaties:

* Verminder SMI interrupts (PS/2 muis/keyboard (geen USB, uitzetten in BIOS))
* Compileer een ACPI-enabled Kernel
* Zet TCO timer genereation van SMI’s uit
* Verminder DMA bus apparaten (SATA/PATA/SCSI, network adapters, hdd en GPU)
* Zet energiebeheer op maximale prestaties
* Zet hyper threading uit
* Zet CPU scaling uit
* Zet CPU sleep uit
* Zet VGA-console uit
* Verhoog de proces-prioriteit en affiniteit.’

Ook kunnen we ervoor kiezen om een RTOS te gebruiken. Deze zouden een hard real-time implementatie mogelijk moeten maken. Wat in principe raar is aangezien volgens de bron de hardware interrupts/BIOS instructies niet aangepast kunnen worden door het OS.