LEAN ROBOTICS

The Top 5 Cobot KPIs

HOW TO MEASURE AND IMPROVE THE PERFORMANCE OF COLLABORATIVE ROBOTS

ROBOTIQ

UNIVERSAL ROBOTS

TABLE OF CONTENTS

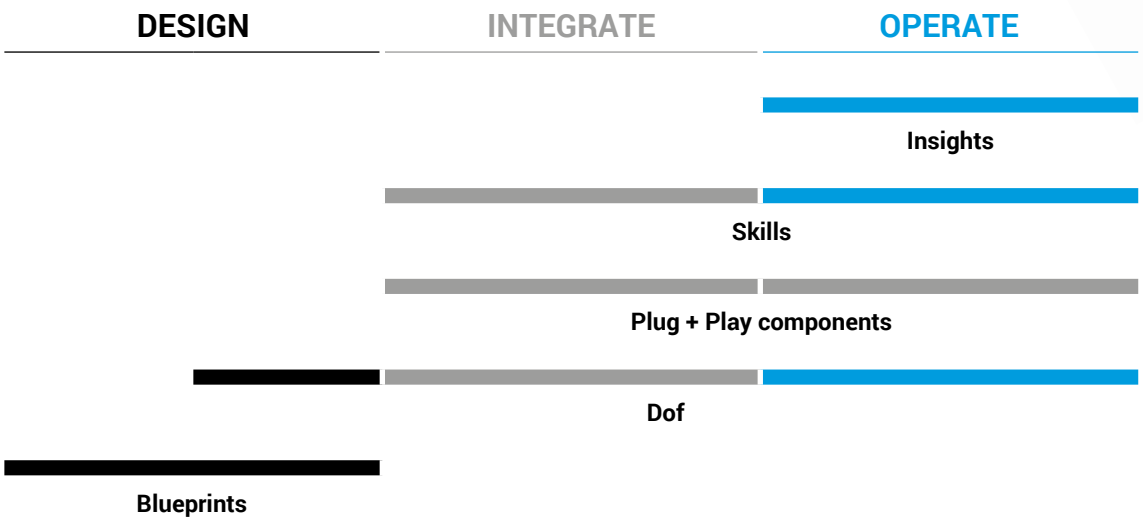
Lean Robotics:
Simplify Robotic Cell Deployments

Whenever you ask if robots could work in your factory, the answer you receive is always a hesitant "It depends." It depends on your factory, your team, which robot you choose, what you want it to do… and a whole lot more.

If you're a first-time robot user, how can you get started? How do you get from your initial idea to a productive, working robot? And if you've already got a few robotic deployments under your belt, how can you scale up your robotics efforts throughout your factory—or across multiple factories?

The answers can be found in lean robotics: **a methodology for simplifying robotic cell deployments.**

Lean robotics is a systematic way to complete the robotic cell deployment cycle, from design to integration and operation. It will empower your team to deploy robots quicker and more efficiently than ever before. Lean robotics divides robotic cell deployments into three phases: Design, Integrate and Operate.



Robotiq's library of eBooks covers the different phases of the robotic cell deployment to ensure that you have access to tips from robotics experts throughout your automation process.

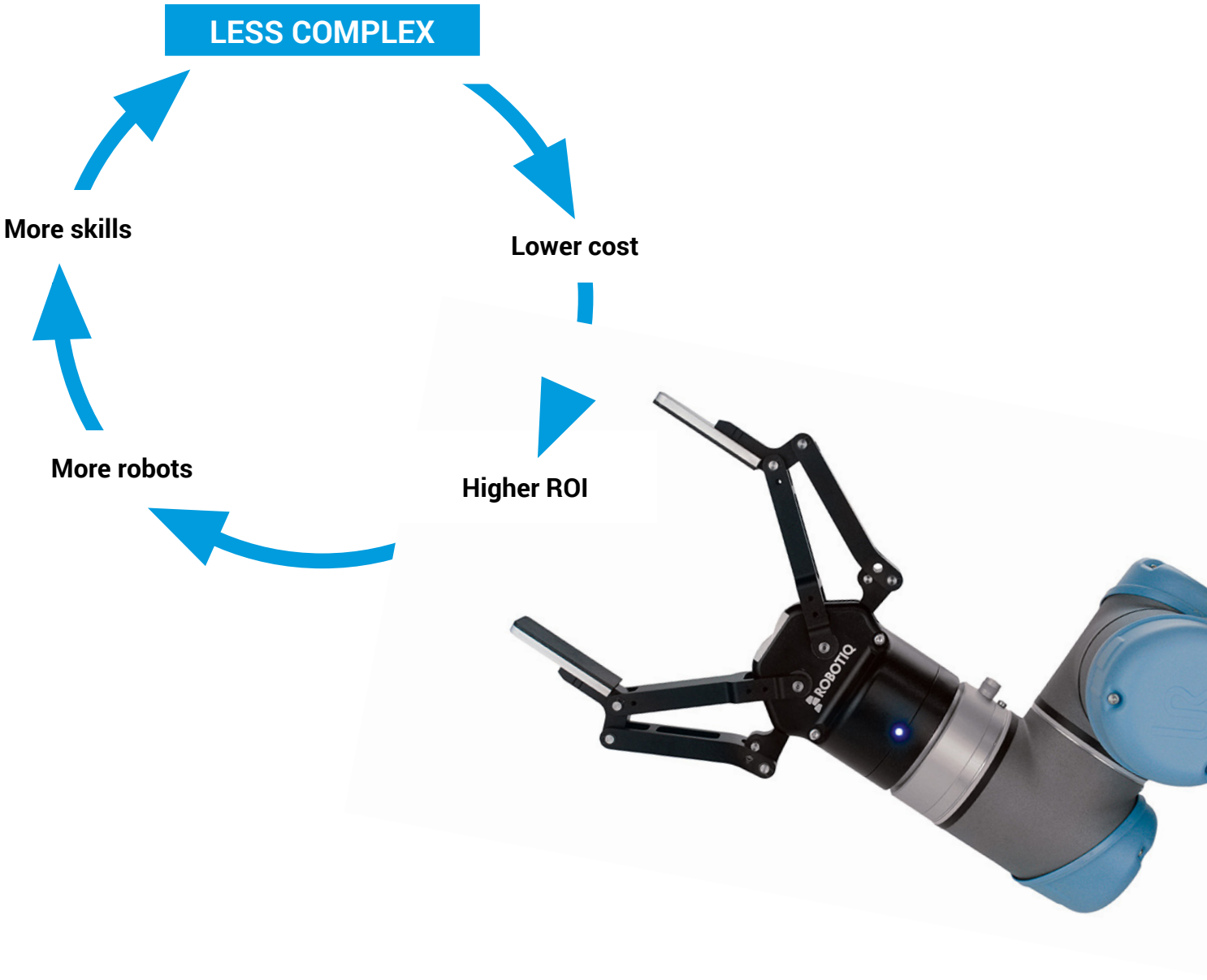
Learn more about Lean Robotics at **leanrobotics.org.**

This eBook Covers the Operate Phase

The operate phase represents the end goal of deployment: having a productive robotic cell that does its job properly on an ongoing basis.

OPERATE

When you're in the operate phase, your robotic cell is finally producing valuable parts for your company, and all your hard work will start to pay off. Since the operate phase is a continuous loop, there are many ways to optimize the performance of your robotic cell (and your plans for the next one).



Introduction

Want to improve the productivity of your collaborative robot?

If you have been using cobots for a while, you might find yourself in one of these situations:

- The robot is performing well, but it is not as productive as I predicted… and I don't know why!?
- I feel that I could boost the robot's productivity further, but I don't know where to start.
- My overall throughput is not great but I don't know how to tell if the robot could help to improve it.
- I don't have time to solve all the problems in the process and optimize the robot at the same time!

Like every machine, collaborative robots are most effective when you use them to their full potential.
But how do you know if you are using a robot to its full potential or not?

This eBook shows how to use cobot-specific KPIs to measure the robot's performance. You can then use the data that you gather to improve the performance of your robotic cell and the manufacturing line as a whole.

ROBOTIQ

UNIVERSAL ROBOTS

What Are KPIs?

You're probably familiar with the concept of Key Performance Indicators (KPIs). Perhaps you already use them to monitor the performance of your manufacturing line.

KPIs are objective metrics for the success of a process. By tracking KPIs over time, you can evaluate which parts of the process work well and which could benefit from further improvement.

There are hundreds of KPIs. Here are some of the best-known ones in manufacturing:

- **Manufacturing Cycle Time**—Time taken to produce a product, from order to release.
- **Yield**—Number or percentage of products that are manufactured to specifications without reworking or scrap.
- **Customer Rejects and/ or Returns**—Number or percentage of products that are returned due to quality issues or customer dissatisfaction.
- **Manufacturin Cost as % of Revenue**—Percentage of revenue that the business spends on manufacturing its products.
- **Downtime**—Amount of time (often expressed as a percentage) that machines in the manufacturing process are unavailable for production.

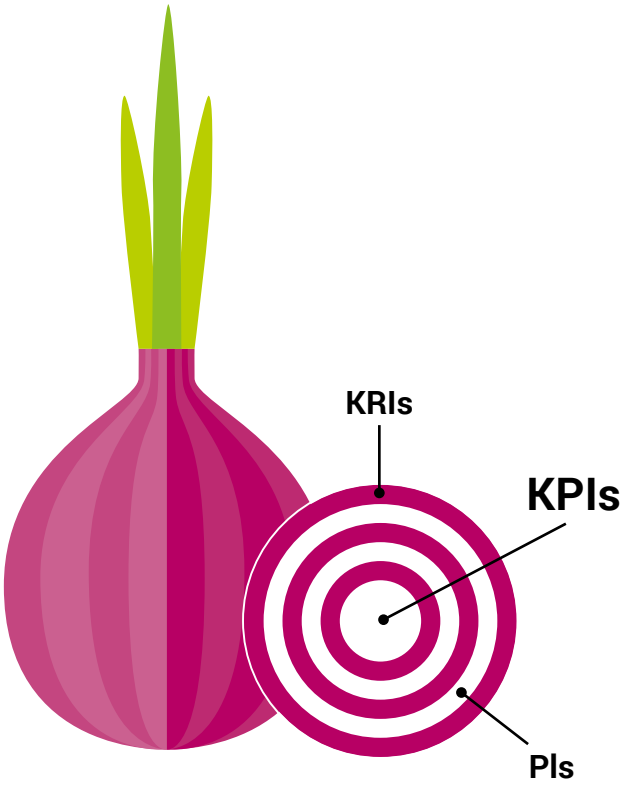
But which KPIs are most important for collaborative robots?

The challenge is to choose only the KPIs that will help you improve your process and add value to your products. Picking too many can be worse than picking none at all.

Why Not All Metrics Are Equal

There are many metrics you could use to assess your manufacturing processes. However, not all of them can be considered key performance indicators, and not all of them will be helpful for collaborative robots. Three different types of metric are used in business. KPI expert David Parnmeter explains them using the analogy of an onion:

- **Key Results Indicators (KRIs)**—These are the skin of the onion. They tell you how the business is doing on the surface, such as net profits, return on capital, and customer satisfaction. KRIs provide a rough indication of how well you've done in the past, but they don't help identify which actions you should take to improve in the future.
- **Performance Indicators (PIs)**—These are just below the surface of the onion. They do give an indication of actions you could take, such as net profit for a specific product line, profitability of a particular subset of customers, or number of training days per month. Your business will have numerous PIs that compliment the KPIs.
- **Key Performance Indicators (KPIs)**—These are at the core of the onion. They are special case Performance Indicators that have the potential to dramatically improve your business's performance. When you choose the right KPIs, you will see huge changes.



But… How do you pick the right KPIs?

First, you have to decide which type of KPIs to focus on.

Strategic KPIs vs. Operational KPIs

A lot of KPIs are designed to measure the performance of a business at the broader management level. These are known as Strategic KPIs and they're not so useful for measuring the performance of machines.

To assess shop floor technology we want to use Operational KPIs, such as overall equipment effectiveness (OEE), machine uptime ,and product quality. These will help you improve the manufacturing process at a lower level.

The two types of KPI are connected. You should be able to link the Operational KPIs to your Strategic KPIs and big-picture goals.

For example, if the business's goal is to increase profits, return on capital might be an important Strategic KPI and machine uptime could be a suitable KPI to choose. However, if the goal is to increase customer satisfaction, cycle time or product quality might be more important, depending on the needs of the customers.

How Do You Apply KPIs to Robotics?

How do robots fit in?

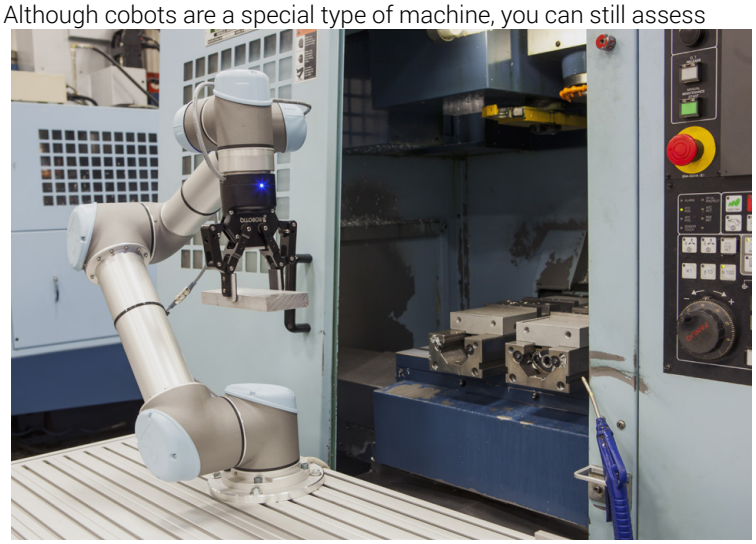
Are they just like any other machine?

Manufacturing machines are often assessed using the Overall Equipment Effectiveness (OEE) metric, which we'll talk about in the next section. This metric can be used for collaborative robots as well. However, it does not indicate how to improve the performance of the robot.

To get the most from your collaborative robot you'll need some cobot-specific KPIs.

KPIs FOR COLLABORATIVE ROBOTS

Before we introduce the five KPIs themselves, let's look at how machines are usually measured in manufacturing businesses. Although cobots are a special type of machine, you can still assess your robotic cell in a way that's similar to your other pieces of manufacturing equipment. Just keep in mind that robotic cells are subject to similar losses as other machines, but there are some additional cobot-specific losses that affect their performance.



How can I tell if a machine-tending robot's running at optimum efficiency?

OEE: The Standard Metric

Overall Equipment Effectiveness (OEE) is the standard for measuring manufacturing productivity. The usual definition is:

OEE = Availability x Performance x Quality Rate

These variables are calculated as follows. Note that the defition of "Count" will depend on your business. For example, Total Counts could be Total Parts Made, Total Products Tested, etc.

$$\text{Availability} = \frac{\text{Actual Run Time}}{\text{Planned Production Time}}$$

$$\text{Performance} = \frac{\text{Ideal Cycle Time x Total Counts}}{\text{Run Time}}$$

$$\text{Quality} = \frac{\text{Number Good Counts}}{\text{Total Counts}}$$

Why OEE Is Not Enough for Collaborative Robots

OEE is a useful metric. However, when used alone it does not provide enough information to help you improve your cobot setup. Here are a few reasons why:

- First, OEE is relatively abstract. It only provides a bird's-eye view of the process and doesn't indicate specific actions you could take to improve your cobot setup.
- Often, OEE is calculated using "planned" or theoretical values. In this case, you don't get a reflection of the real performance achieved at specific stages of the process.
- OEE was designed to deal with "major losses." Many of the improvements you can make with a collaborative robot are small but they have a big effect on productivity over time. OEE may not take these smaller losses into account.

OEE only tells part of the story. It's well-suited to measuring the effectiveness of the entire robotic cell, but it doesn't go into enough depth to indicate the effectiveness of the robot alone.

For example, consider a machine-tending cobot.The cobot itself doesn't perform value-added operations on products—that's the CNC machine's job. So how can you tell whether the cobot's running at optimum efficiency?

With OEE, you would only be able to calculate efficiency for the paired cobot and CNC machine setup. You need a way to quantify the efficiency of the robot alone.

In order to get the most from a cobot, you have to "zoom in" on its performance. For that, you need some cobot-specific KPIs.

Which KPI Properties Are Important for Cobots?

The ideal collaborative robot KPIs have the following characteristics:

- **Non-financial**—While it's certainly important to calculate the Return on Investment (ROI) of your collaborative robot, a metric like this won't tell you how to improve its operation.
- **Measured continuously**—Cobots make it easy to gather data, since they're able to log it directly within their programs. It's a good idea to log your KPIs continuously, because then you can compare both long-term and short-term data. This makes it easy to quantify the effect of small changes on the robot's performance.
- **Linked with other Operational KPIs**—The robot's operation will affect other KPIs within the business (e.g. time from order to shipment, manufacturing cost per unit, plant downtime).The metrics you use for the cobot should have clear links to broader effects on the business.
- **Focused on one or more of the common losses**—Many of the performance gains in collaborative robotics can be achieved by tackling some common losses. KPIs that reflect these losses will likely point to ways of improving performance.
- **Easy to measure**—KPIs that are hard to measure won't be measured at all. When picking a KPI, ask yourself how much time it will take to gather the data, then consider whether you'd be happy to spend that time when you're very busy. If not, it's not a good KPI.
- **Clear and simple**—The best KPIs can be understood without any extra training. The simpler they are, the more useful they'll be for everyone when it comes to optimizing the process.

10 Common Losses in Cobot Operation

OEE is often associated with ten big losses, which also apply to robotic cells:

1. **Planned downtime**—Due to changeovers, planned maintenance, end-of-arm tooling changeover, etc.
2. **Breakdowns and unplanned downtime**—Due to equipment failure, unplanned maintenance, etc.
3. **Minor stops**—Due to misalignment, blockages, safety stops (e.g. due to people entering the workspace), etc.
4. **Speed loss**—Due to untrained operators, inefficient waypoint programming, misalignment, etc.
   This category is a major loss for some collaborative robots, which automatically enter a "reduced speed" mode when people enter the workspace or touch the robot. Monitoring and minimizing such events (e.g. by teaching people to only enter the workspace when necessary) is an easy way to improve performance.
5. **Production rejects**—Due to damaged products, scrap, etc.
6. **Rejects on start up**—Due to scrap caused by changeover, damage, etc.

There are also four additional losses that affect collaborative robots in particular:

7. **Integration faults (or communication faults, for cobots)**—The cobot sufers from brief stoppages due to faults such as errors in machine-to machine communication or synchronization, misaligned parts, connectivity failure, etc.
8. **Lack of use**—The cobot is not being used to its full potential due to lack of cobot training, process optimization issues, poor resource allocation, etc.
7. **Inefficiency (or: poor trajectory planning, for cobots)**—An unoptimized cell layout leads to inefficient movement of the robot. The robot's paths should be measured and optimized to eliminate this waste.
10. **Wait time**—The cobot is unable to achieve its full potential because it is waiting for other processes. This can be due to limited understanding of the cobot's full potential, bottlenecks or unoptimized steps elsewhere in the process, etc.

Ideally, every KPI you choose should address one or more of these common losses. All five of the Cobot KPIs introduced in this book address several of these different losses.

Get More from Cobots with Lean Robotics

You can apply Cobot KPIs to a robotic cell without making any other changes in your business. However, you may wish to supercharge your cobot by using KPIs as part of the Lean Robotics framework.

Lean Robotics is a method for efficiently incorporating robots into your business. Based on Lean Manufacturing principles, it guides you through the entire robotic cell deployment process, from design to operation.

Lean Robotics covers three phases:

1. **Design**—Completing the robotic cell plan and comparing the plans with the existing manual cell.
2. **Integrate**—Installing the cell and completing high-level robotic programming.
3. **Operate**—Operating and continuously improving your robotic cell.

Cobot KPIs are most useful during the final phase, Operate, because you use them to measure the performance of the robot and continuously improve its operation.

To learn more about Lean Robotics go to **leanrobotics.org** and get a copy of the book.

ADDING MORE KPIS USING THE NEXT-CELL-AS-A-CUSTOMER MINDSET

In this eBook, we'll introduce the five most important Cobot KPIs. By mastering these metrics, you can track huge improvements in your manufacturing processes. However, there's a way for you to apply even more metrics to your collaborative robot—by using the mindset of "next-cell-as-a-customer," also known as the "internal customer" mindset. This is another key part of the Lean Robotics methodology.

The approach involves thinking of the robotic cell as if it were its own self-contained business. Imagine the next cell in the process is the robotic cell's customer, and the previous cells are its suppliers. Using this mindset, you can apply business-focused KPIs to your robotic cell (e.g. customer rejects, on-time delivery to commit, supplier's quality incoming, etc.). In practice, the five KPIs in this book should be more than sufficient for most people, but this approach opens up many possibilities for further optimizing your robot.

For more information on applying the next-cell-as-a-customer mindset visit **leanrobotics.org**.

How to Record Cobot KPIs

It can take a surprising amount of effort to gather data on any KPI, no matter how simple it seems at first.

For example, imagine your company wants to measure the number of non-compliance events per year (the number of products that don't meet specifications). During a meeting, management decides they want to track this metric. At first, the solution seems simple: just count all the products that fail a quality check.

However, putting it into practice isn't so easy. There are several operational decisions to be made:

- **Who will record the KPI?**—Will it be done by machine operators or recorded automatically? In the former case, the operators may need training. If it's the latter case, who will make sure you have all the equipment needed to automatically measure non-compliances?
- **How often should the KPI be recorded?**—If machine operators are required to log all non-compliant products during their shifts, it could affect their output. On the other hand, if they only measure it every so often, the KPI will not present an accurate measure of productivity.
- **What else needs to be recorded?**—Will you only count the number of non-compliance events, or should operators also record the reasons, times and resolutions of those events? How much time will this take away from their production-related tasks?
- **How will we analyze the KPI?**—Who will be responsible for the analysis? How much time will this take away from his/her other responsibilities? Is any new software needed to perform the analysis?

When you add up all the time and costs associated with logging KPIs, you might be surprised how much work is involved in a simple idea like "count the number of defective products." This is why it's important to carefully choose the logging methods you will use to gather KPI data.

There are at least four ways to track KPIs, each of which involves different amounts of time and effort.

Manual Tracking

The simplest way to track most KPIs is by manual tracking. In the past, this was the only method. It involves recording the data during one's shift, either on paper or digitally, e.g. with a smartphone. The responsibility for data-logging usually falls on the shop floor staff.

Manual tracking has some advantages:

- It's easily understood by everyone involved.
- It doesn't require any special software. A notebook and pen for logging and spreadsheet software for analysis is often sufficient.
- It's (relatively) cheap and quick to implement. It doesn't require investing in new technology, skills or business processes, and as long as it doesn't need much training, KPI logging can begin within a few days.

However, manual tracking also has some major disadvantages:

- It's time-consuming, both for the staff who are doing the logging and for whoever has to input and analyze the data afterward.
- It's often inaccurate. It relies on operator diligence so it's common for people to forget to log the data when things go wrong in the process. Unplanned downtime, for example, is a typical case where data would be useful after the event but is unlikely to have been recorded during it.
- It's unnecessary for most robotic applications, since robot controllers can be programmed to log metrics automatically.
- It detracts from the job of production, which can be costly in the long term.

Simple Controller Logging

It is possible to program a robot's controller yourself to log some metrics automatically. This involves writing a subroutine to record whenever a particular event occurs. For example, if you want to record how many times a machine tending robot presses the Start button on a CNC machine, you could make a variable called "cnt_starts" and program the controller to increment the variable every time the start button is pressed.

Simple controller logging has some advantages:

- It is easy to implement if you or a member of your team is proficient in robot programming.
- It is completely customizable to the needs of your application.
- It does not have the same problems that manual tracking does, as the robot will always log the metrics.

However, simple controller logging also has some disadvantages:

- It requires at least one member of your team to have the ability to program the subroutine and subsequently download the information from the robot when it is time to analyze the data.
- You don't receive real-time updates from the robot. At its most basic, this method only records information in the robot controller. Of course, it's possible to program the robot to send the information across a network, but this increases the complexity of an otherwise simple programming task.
- It can lead to inconsistency and inaccuracy. Imagine two engineers program two cobots separately. Each one may program their robot to log a metric in a slightly different way, which can lead to incomparable data that's hard to properly analyze.

Cobot Integration with MES

Manufacturing Execution Systems (MES) software is a type of enterprise solution for managing factories. In manufacturing environments, it has traditionally been the main option for advanced logging of business metrics. The software keeps track of manufacturing processes in real time and is able to monitor KPIs over time.

MES software has several advantages:

- It's software-based so there's no need to log information manually.
- It's easy to analyze the data and share information company-wide because it's all stored within the system.
- It provides a huge amount of data, enabling you to (potentially) get to the root of many problems.

However, MES software also has some disadvantages for collaborative robots:

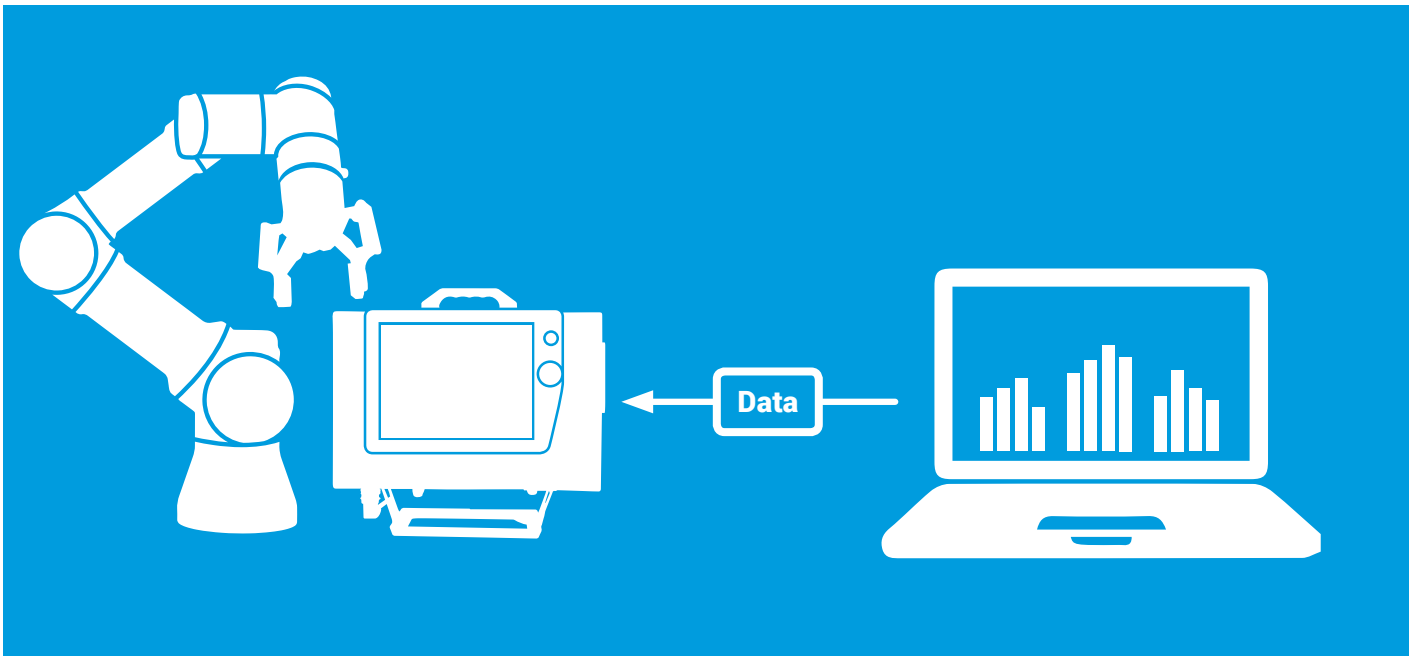
- It is not integrated with collaborative robots out-of-the-box. Using cobot data with an MES would require someone to program the robot controller to send the data automatically to the MES system. This level of programming would require an experienced robot integrator.
- It's costly and requires a business shift. MES software solutions are a huge investment, so if you're not using one already, implementing one can require a change of business culture.
- It only looks at the big picture. Like the OEE metric we discussed previously, MES software tends to focus on the "big picture" KPIs of manufacturing processes. Although it is possible to track any metric using MES software, this tradition means that most users use them to track the principal machines (e.g. CNC machines) and not the robots.

Cobot-Specific Monitoring Software

A third option is to use a solution specifically designed for monitoring collaborative robot KPIs. This software communicates directly with the robot's controller and records the data in real-time. It includes integrated visualization methods, and the data can be exported for further analysis.

Cobot logging software has some advantages:

- It's quick and easy to implement because it only requires a single piece of software. Unlike MES systems, it is not a business-wide solution so it doesn't require nearly as much of an investment.
- It's designed to get the most from cobots. By tracking cobot-specific KPIs, this software allows you to maximize your use of cobots—unlike MES software, which traditionally focuses on the "big picture" of the manufacturing process.



- It provides data in real time. As opposed to with manual tracking, the data are automatically sent from the robot controller to the analysis software. Since no user input is required, they (the data) can be viewed immediately.

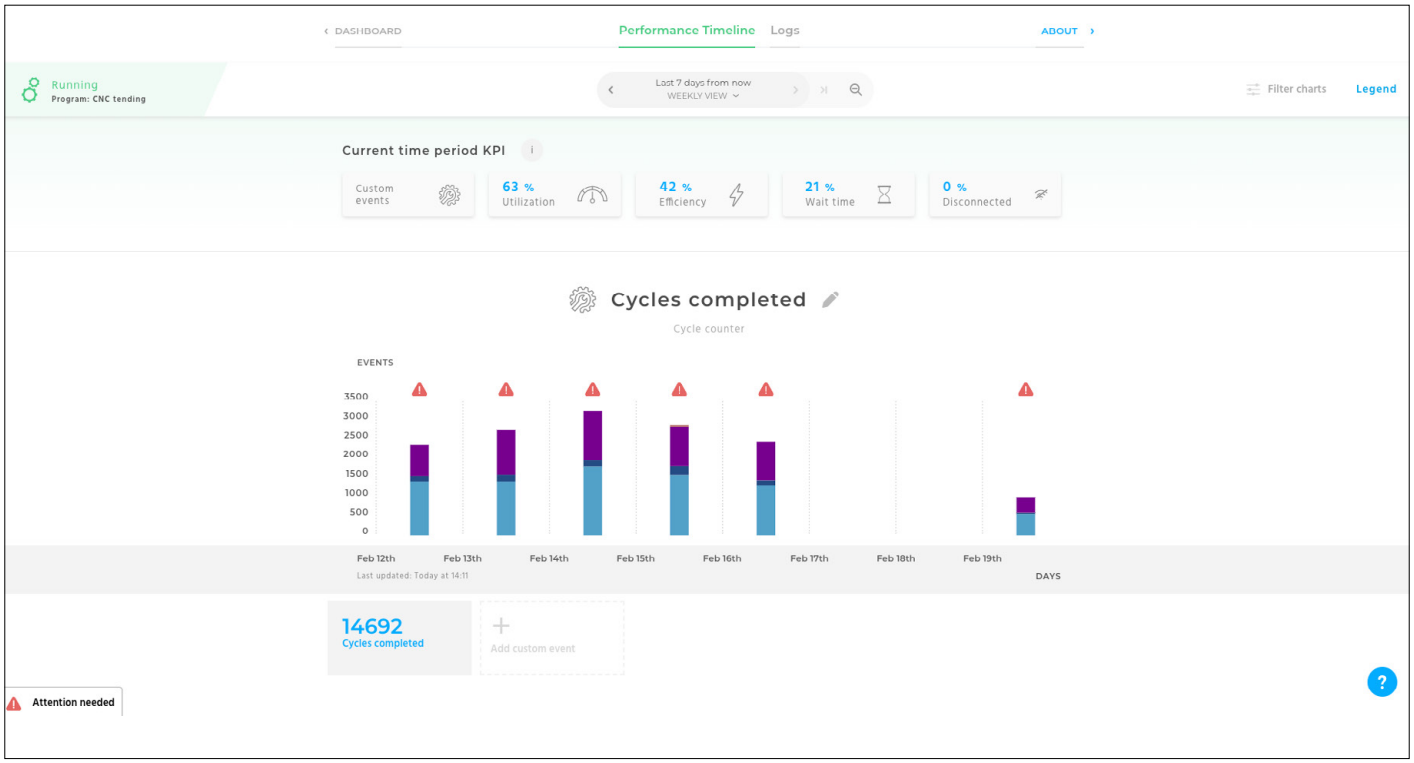
Cobot logging software also has some potential disadvantages:

- It requires a cobot. Because it's a specific software solution, it won't interact with any non-cobot machines within your process. Also, its compatibility may be limited to certain brands of robot.
- It is a separate software solution. If you are already using MES software you may be less inclined to add yet another stand-alone software into your business. Although the data can be exported from cobot software and imported into the MES software, someone may have to do this manually or write custom code to export it automatically.
- It only logs cobot KPIs. Because it focuses on the performance of the robot itself, it does not natively track other business KPIs, such as customer satisfaction or quality. These would have to be analyzed separately.

How To Find Out About Cobot-Specific Monitoring Software

One example of such software is **Robotiq Insights**. It includes a cloud-based software package that's built for easy integration with Universal Robots.

Insights automatically logs the top five cobot KPIs in real time, and allows you to monitor several robots at the same time. Here's an example of the type of interface you can see in the software:



Monitoring KPIs Over Time

A major advantage of centralized, sotware-based solutions (i.e. a cobot-specific monitoring software or a well-configured MES) is that they allow you to easily monitor KPIs over time and with different robot applications. This is important because KPIs can vary greatly depending on shifts, operators, products, etc.

Manual tracking can only ever provide a brief snapshot of manufacturing performance; it doesn't let you see the overall picture. Simple controller logging can provide data about the robot's overall performance, but usually lacks the ability to zoom into the data in detail.

THE PROBLEM WITH INFREQUENT, MANUAL LOGGING

Imagine that you manually record your robot's KPIs at 11:00 am every Monday morning.

At that time, the robot is usually running a packing application. You record it running through a few cycles and calculate that it's running at 90% Efficiency, 100% Utilization, 5% Wait Time, and 120 Cycles Completed per hour.

Is this an accurate measure of the robot's overall performance?

Of course not! It is accurate at the exact moment that you make the measurements, but it's probably inaccurate at all other times. It only tells you how the robot performs for a packing application at 11:00 am on a Monday.

How well does the robot perform on the other days of the week? At different times? For different applications? You don't have enough data to be able to tell.

This is a major issue with manual logging of robot KPIs. Of course, some data is (usually) better than none. However, such infrequent logging is of limited use.

THE PROBLEM WITH SIMPLE CONTROLLER LOGGING AND MES

Simple controller logging usually has the opposite problem: it doesn't provide enough detail.

Imagine that you program the robot to increment a variable every time it picks and places one object. You start the robot running at 9 am and come back at 3 pm to move it to another task. You have programmed the robot to output the value of the variable to the screen when the program ends. It tells you that the robot has picked and placed 952 objects during the six hours.

Is this an accurate measure of the robot's overall performance? Yes. However, it doesn't tell you any details about the robot's performance.

Was the robot's performance consistent every hour? Or did it move more objects at particular times?

Did the robot stop for any reason during the six hours?

You could add more logging functionality into the robot's controller, e.g. by incorporating timestamps and a file I/O into the program. However, this would add more complexity to the program. It would also be more complicated to analyze the data later. This is also the problem with MES sofware, as you would have to program the logging routines yourself into the robot controller.

THE BENEFIT OF MONITORING TIME FRAMES AND TASKS

Cobot-specific monitoring software provides one huge benefit for KPI analysis: it can easily compare different time frames and tasks, allowing you to obtain a comprehensive view of the robot's performance. The more data points you have, the better you will understand your robot.

Robotiq Insights, for example, presents data over the time frames of a month, week, day or hour. It differentiates between tasks by storing KPIs individually for each program.

This allows you to answer questions that would otherwise be difficult or impossible, such as:

- How effectively do employees use the robot during various shifts?
- What are some differences in how various teams use the robot?
- Which tasks take up most of the robot's time in the long-term?
- Which tasks are consistently inefficient?

Comparing different time frames and tasks allows you to identify patterns and pinpoint problem areas, helping you tackle recurring issues.

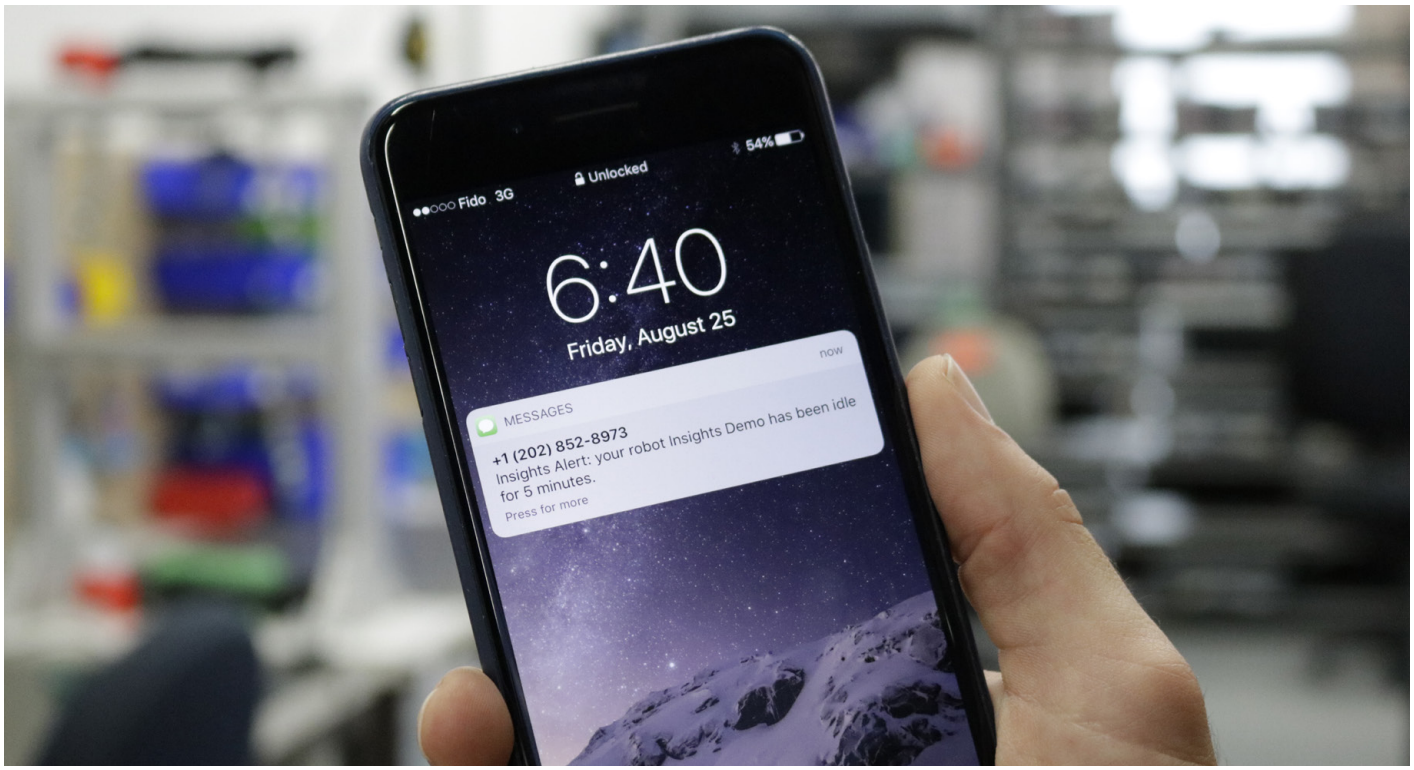
For example, you might look at the monthly view and find that the robot has unusually low Utilization every Thursday. You could then use the daily and hourly views to identify the root of problem: the robot does not run a program until 1:00 pm every Thursday. After talking to members of your team, you may find that no member of the morning shift knows how to use the robot, so it sits idle until the afternoon shift begins. This is an easily solvable problem that might otherwise have gone unnoticed.

This type of analysis is simple using cobot-specific monitoring software, but it would be unwieldy with any of the other recording options. Manual logging would have to be done near-constantly to achieve the same level of detail, which is highly impractical. Even though MES systems can track KPIs over time, they cannot easily differentiate between robot tasks and would require manual entry of most cobot KPIs.

IMMEDIATE NOTIFICATION

Another benefit of monitoring software is that you can immediately tell when a robot has encountered a fault or entered an undesirable state.

Robotiq Insights incorporates a notification service that sends SMS messages to notify you when the robot enters any of the following states:



- **Emergency Stop**—This state stops the program and cuts power to the robot. It's usually triggered by an external signal like an emergency stop button.The robot must be manually reset to clear this state.
- **Safeguard Stop**—Like the Emergency Stop, this state means that the robot has stopped following an external signal. However,it only pauses the program and there is power to the robot. The program will continue either automatically or following a manual reset, depending on the robot's configuration.
- **Protective Stop**—This stop state is triggered by the internal safety limits of the robot control system. It can only be reset manually.
- **Idle**—The robot is powered up but no program is running. See the later section on Utilization.
- **Disconnected**—The robot is powered down or otherwise disconnected from the network. See the later section on Disconnect Time.

Notifications are a major factor that differentiates "monitoring" software from basic "logging" software. Cobot-specific monitoring software provides tools that allow you to respond to the robot in the moment as well as improve your long-term analysis of KPIs. As you read the following sections of this eBook, consider the benefits that monitoring each KPI over time could bring to your use of cobots.

Top 5 KPIs for Collaborative Robots

Here at Robotiq, we have a decade of experience working with robot users from a variety of industries. We've come up with five KPIs that we find are most useful for improving a collaborative robot setup.

Here are our Top 5 KPIs along with detailed guidance on how to use them.

KPI 1. Cycle Time

Cycle Time should be familiar to you if you have experience in manufacturing, as it is a commonly used metric at several levels of manufacturing businesses. For example, "Overall Cycle Time" is used to measure how long it takes for a product to pass through the entire manufacturing process, whereas "Machine Cycle Time" measures the processing time of a single machine.

Traditionally, Cycle Time measures how long it takes from the moment a product enters a process to the moment it exits. For cobots, however, Cycle Time measures the duration of one robot sequence. Unlike the traditional definition, this means that it does not necessarily indicate how many objects were processed, as you might process multiple products per sequence.

We define Cycle Time as:

Cycle Time = Time Sequence Starts – Time Previous Sequence Started

In general:

- **A short Cycle Time is optimal**—Shorter Cycle Times are better because they mean you can process more products in a given time period.The shorter the Cycle Time, the more optimized your robot is.
- **A long Cycle Time is undesirable**—Longer Cycle Times mean you can process fewer products in a time period. Although some operations may just take a long time, there are often ways to optimize them.

HOW CYCLE TIME APPLIES TO COBOTS

Cycle Time can be considered the basic metric for improving the performance of your collaborative robot. When you push down the Cycle Time of each step in your process, the overall Cycle Time is reduced as well.

The main difference between collaborative robotic cells and other steps in your process is that robotic cells are usually autonomous. As a result, the Cycle Time time for a robot is likely to be more uniform than for fully manual processes. Despite this difference, you should use Cycle Time in the same way you would for a manual process: by making small changes to the robot's operation to save time.

Cycle Time is a bit different from the other KPIs in this eBook, as it is most useful when you sit down to analyze the performance of your robot. The other four KPIs are also useful as real-time performance metrics, to warn you when a robot has stopped performing as well as it could.

HOW TO CALCULATE A TARGET CYCLE TIME
The goal is to have short Cycle Times as often as possible.

To achieve this goal in your robotic cell, it's advisable to set a "Target Cycle Time":

Target Cycle Time—The fastest repeatable time required to process a single object. This becomes the benchmark for the process.

Note the requirement that the Target Cycle Time is "repeatable." Even when you are using robots, the Cycle Time will never be exactly the same between cycles. Sometimes you will have an unusually short Cycle Time, and other times a cycle will take ages (e.g. due to a breakdown). It can be tempting to take the shortest time as your Target Cycle Time — to try and improve the performance of the cell by settng a high standard. However, this is unrealistic.

The Target Cycle Time for each step in your process should be the average of all of the times you measure for that step.

But what type of average should you use? That depends on the properties of your measurements.

There are three ways to calculate the average Cycle Time: mean, median, and mode. Here's how to choose the method that's best for you:

- **Mean Cycle Time**—If the measurements are all similar to each other with no outliers, use the mean. Sum all of the values together and divide by the number of values:

$$\text{Mean}_n = \frac{X_1 + X_2 + X_3 + \ldots + X_n}{n}$$

- **Median Cycle Time**—If some measurements have high or low outliers, use the median. Arrange the values in ascending order, then pick the value that's precisely in the middle of the range:

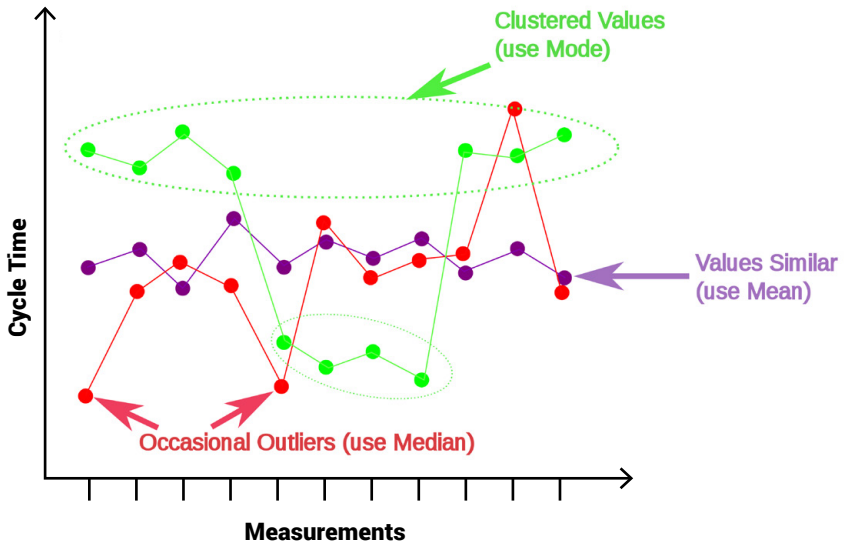
$$\text{Median}_n = \frac{(k+1)^{th}}{2} \text{ value in ordered set of k values}$$

- **Mode Cycle Time**—If your measurements cluster into two or more sets of values, use the mode. Take the value which appears most often as the average.

The value you calculate will become your Target Cycle Time.

6 FACTORS THAT AFFECT CYCLE TIME

1. **Cell Layout**—Try to shorten the distance that the robot has to move between each step of the process. Keep areas close together, especially if the robot will be moving between them regularly.
2. **Robot Motions**—Cut any unnecessary movements from the robot's programming. Ask questions like: "Is it really necessary to use five waypoints for this motion or will two be sufficient?"
3. **Robot Speed**—Increase the speed and accelerations to the maximum values that make sense for the application, keeping in mind any collaboration-safe limits.
4. **Combine Operations**—Look for any steps of the task which could be doubled up or removed. (E.g., ask "Can I open the gripper while the robot is traveling instead of waiting until it has reached the pick-up point?")
5. **Grasp Time**—The time it takes to open and close the robot gripper can have a significant effect on cycle time. Make sure it's optimized.
6. **Processing Times**—The robot is often working in collaboration with a CNC machine, human operator, or some other part of the process. If this other part causes delays, it will affect the robot's Cycle Time.



However, don't forget about your outliers! If you used the median or mode, the outliers can provide useful information about how you can improve the performance of your cell:

- **Every time you see an outlier, ask yourself:** What happened during that cycle to increase or reduce the Cycle Time?
- **For each cluster of values, ask yourself:** What's the difference between the cycles in each of these clusters?

COMMON LOSSES WHICH AFFECT CYCLE TIME

Cycle Time is negatively affected by a few of the common losses:

- **Speed loss**—If the robot has been programmed with unnecessary waypoints, or doesn't achieve its full speed due to being unnecessarily set in a "reduced speed" mode, it will take longer to travel around the workspace and the Cycle Time will be longer.
- **Minor stops**—Any stops, even those that do not require halting the robot's program, will increase the Cycle Time, and often show up as outliers.
- **Faults**—Regular faults in the robot's programming will consistently increase the Cycle Time. Occasional faults will cause outliers.
- **Inefficiency**—If the robot has been programmed by an untrained operator, there could be many small inefficiencies in its operation that will increase the Cycle Time.
- **Wait Time**—Waiting for other processes to finish can significantly increase the Cycle Time.

HOW TO MEASURE ROBOT CYCLE TIME

There are at least five ways to calculate the Cycle Time of a collaborative robot:

1. The Imprecise Way

You might think you could calculate the Cycle Time by doing the following:

1. Go to the robotic cell.
2. Count the pile of processed objects stacked up at the end of the cell.
3. Divide the total shift time of the robot by this number to calculate the time it took to process each object.

This may be a rough measure of the Cycle Time, but only if the robot only processes one object per cycle. This method is also imprecise because it includes all of the unexpected waits, breakdowns, etc.

If you're lucky and there are no unexpected waits or breakdowns, this value will reflect the mean Cycle Time. However, the mean is an inaccurate measure in this case. It inflates the Cycle Time rather than reducing it. If you were to set it as your Target Cycle Time, it would lower the bar for performance of the cell because you would be aiming for a less-than-optimal value.

2. The Back-of-the-Napkin Way

You can estimate the Cycle Time by using the robot's theoretical speed and making some educated guesses. This is a reasonable way to calculate Cycle Time before you set up the cell – when you will be unable to run physical tests.

To estimate the Cycle Time, work out all of the steps of your robot sequence and assign estimated times to them.

For example, for the machine tending layout shown at right, you might have the following steps:

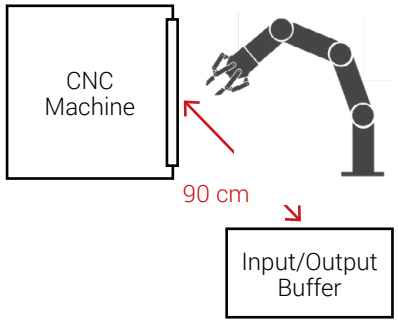
1. Move to buffer (90 cm @ 1 m/s) = 1 s
2. Pick up part from input buffer = 4 s
3. Move to machine (90 cm @ 1 m/s) = 1 s
4. Load part in machine = 8 s
5. Close door = 5 s
6. Press button to start the machine = 3s
7. Wait for machine to finish = 120s
8. Open door = 5s
9. Unload part from machine = 8s
10. Move to buffer (90cm@1m/s) = 1s
11. Drop part in output buffer = 3s

This gives an estimated cycle time of 159 seconds (2.65 minutes).

You can estimate these values by looking at:

- **Grasp time**—Use the gripper opening time equation (Tgripper).
- **Move times**—Take the theoretical speed of the robot and multiply it by the distance.
- **Wait times**—Take the known processing time of any CNC machines or other processes.
- **Other times**—If you have no robot available to test the other times (e.g., how long it takes to open a door) make an educated guess.

An example machine-tending setup:



HOW TO OPTIMIZE THE GRASP TIME

Gripper Margin is the distance needed between the two fingers of a gripper in order to pick up a part. It has a huge effect on the cycle time of the gripper.

For example, it can take 4x longer to complete a grasp just because you chose a gripper margin that was too conservative.

You can calculate the time it takes to open your gripper (Tgripper) using this formula:

$$T_{gripper} = T_{idle} + \frac{2 \times \text{Gripper Margin}}{\text{Finger Speed}}$$

See our **blog post** for more details on reducing the gripper cycle time.

The Cycle Time that you get from this type of estimation is admittedly crude, but it's a start.The advantage is that you can calculate it quickly and without any programming or physical tests.

3. The Time-Consuming Way

The traditional way to measure the CycleTime is to get out your stopwatch and follow these steps:

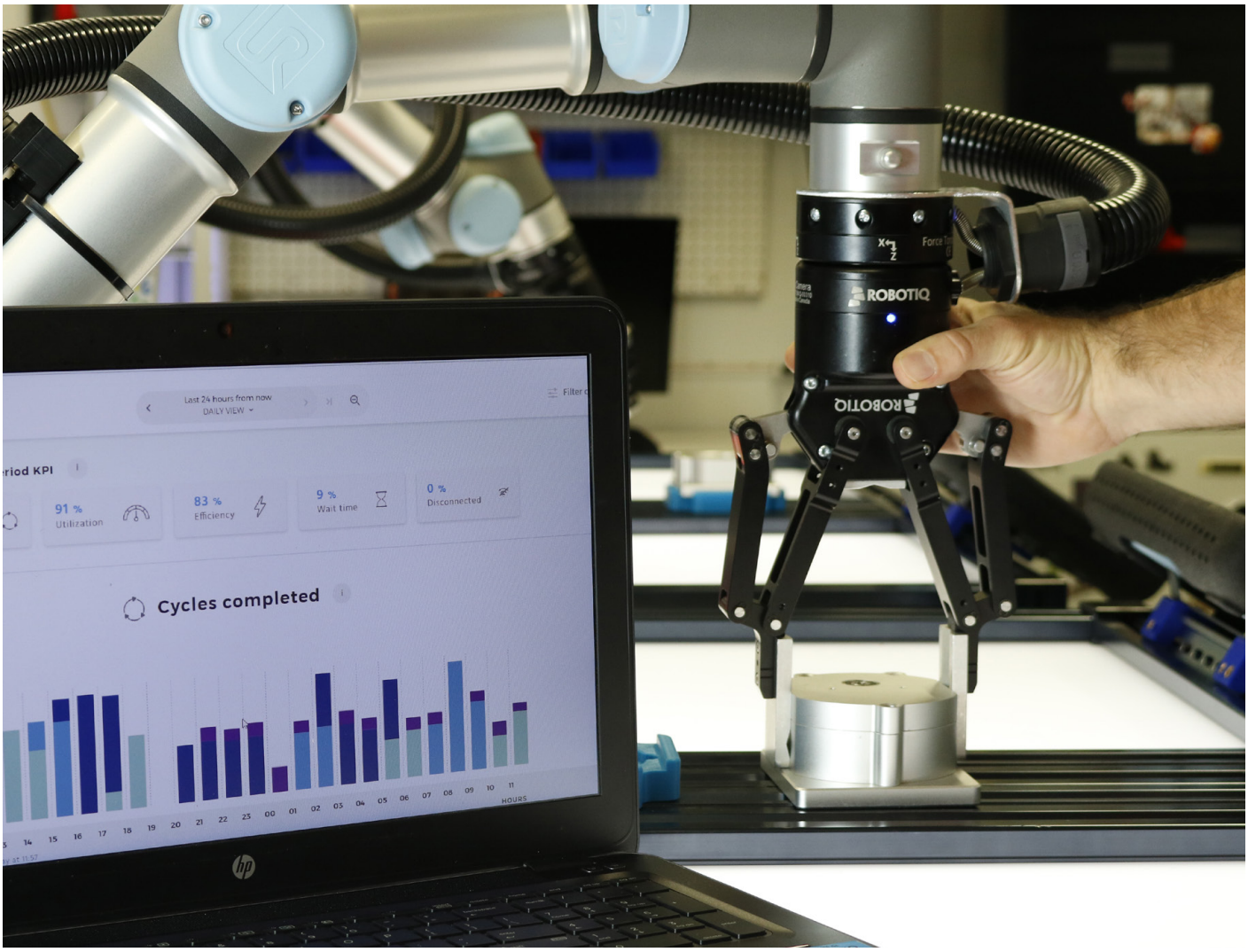
1. Set up the robotic cell that you want to measure.
2. Define the sequence of events for the whole task, like we did in The Back-of-the-Napkin Way.
3. Start the process and measure the times of each stage using the stopwatch. Use a stopwatch with a "lap" timer, because there won't be time to pause, record the time, and restart the clock while the robot is running.
4. Let the robot run through the sequence several times—the more the better.
5. Calculate the average time for each stage using the most appropriate method, as described above.
6. Sum the averages to give an overall Cycle Time.

This provides far more accurate results than the previous two methods. However, it is time-consuming, and as a result you probably won't do it very often, which makes it hard to monitor changes in the Cycle Time over time.

4. The Quick-and-Dirty Coding Way

You can program the robot's controller to record a basic measure of Cycle Time. One way to do this is to save a variable with the start time of the current robot loop. When the next loop starts, use the equation provided above to calculate the Cycle Time of that loop.

This is a useful method during the Integrate Phase when you only want to record one or two cycles. You can output the Cycle Time to the display of the teach pendant and read the values. However, this method can become unwieldy for recording the long-term performance of the robot during the Operate Phase because you have to store all of the data somewhere.



You have at least three options for storing historical Cycle Time in the robot controller:

1. **Store values in an array**—Each time a cycle ends, you store the Cycle Time as a new value in the array. This can be useful when there are a small number of cycles, but unless you have some other way to store the values there is no easy way to record the values outside of the robot controller.
2. **Implement File I/O**—This involves storing the values to a file on the robot controller. It requires extra coding, which can be complex in some programming languages and means you have to copy the data from the robot for analysis.
3. **Send data over a network**—This involves programming the robot to send the Cycle Time over the network, e.g. using network shared variables or sockets. You would also have to write a program on the receiving end to store or process the data.
4. **Take a running average**—A rough measure of average Cycle Time would be to record a running total of all individual times then divide by the number of cycles. This avoids the problem of storing many data points in the robot, but it means that you lose the finer details of the data.

5. The Easy, Accurate Way

The final way to measure Cycle Time is to use cobot-specifc logging software, such as **Robotiq Insights**.This records the movements of the robot continually and automatically stores the start time of each cycle. You can then download this information to CSV and calculate the Cycle Time using your favorite analysis method (e.g. spreadsheets).

You can also assign custom events to individual stages of the task sequence. For example, you could assign a custom event to each of the task steps outlined in "The Back-of-the-Napkin Way" section above. Insights would then log the start times of each of these steps. This is a very powerful feature as it allows you to calculate the detailed Cycle Times of each step of the process, which means you can optimize down to a very detailed level.

KPI 2. Cycles Completed

Cycles Completed shows how many cycles have been performed by the robot in a particular time period. For some robot applications the term "Cycle" isn't the best word to use, as the robot isn't running a cyclic program (i.e. one which restarts as soon as it is finished). In this case, you can use whatever name is best suited to your application, e.g., such as production completed, processes completed, etc.

It's a simple metric but it serves as the cornerstone for other calculations. For example, it can be used to calculate:

- **Yield**—This is the amount of defect-free product that successfully makes its way through the manufacturing process.
- **Throughput**—The amount of product that can be processed in a certain time period. It is closely related to the Cycle Time.
- **Production Target**—This is set as a benchmark for performance of the process. It defines how much product is expected to be produced in a given time period.

HOW CYCLES COMPLETED APPLIES TO COBOTS

Many metrics used in business (particularly in manufacturing) include the number of defects a process produces, e.g. Yield. However, Cycles Completed does not differentiate between good or bad units. One reason for this is that there is no easy way for a robot to automatically detect whether a part is defective. You can estimate yield by detecting if the cycle was successful (i.e. without failures like dropped or misplaced parts) but robots can't detect physical defects, unless they have been integrated with **autonomous metrology sensors**. They usually rely on inspection processes (either manual or automatic) to determine if a product has been manufactured to specification. Cycles Completed is independent of other processes, which is one of the requirements of a good cobot KPI (see the previous section).

HOW TO SET A PRODUCTION TARGET

The Production Target is a benchmark of how much work the robot should complete in a particular time period, e.g. "units per day," "tonnes per hour," etc.

This number shouldn't simply be plucked out of thin air —it should be based on the real capacity of your process.

First, log the Cycles Completed over several hours and calculate the average (using the best method, as explained in the previous section) to give Cycles Completed Per Hour. Multiply this by the Utilization (see next section) to give an Hourly Production Target using the equation:

$$\text{Production Target/hr} = \frac{\text{Cycles Completed/hr}}{\text{Utilization}}$$

Then, use this to calculate a realistic Production Target for your robotic cell. For example, we tested a UR robot in a CNC machine tending task. Over the course of an hour, it completed 21 cycles at 79% Utilization, to give:

$$\text{Production Target/hr} = \frac{21}{0.79} = 26.5$$

We want to run this for 4 hours, so the total production target becomes:

$$\text{Procuction Target} = 4 \times 26.5 = 106 \text{ units}$$

Use this value when you plan the rest of your manufacturing process. Decide if you need to produce more or fewer units to achieve the desired production over a whole day.

Of course, if necessary, you can combine a cobot with external metrology devices to automatically measure the Yield.

COMMON LOSSES WHICH AFFECT CYCLES COMPLETED

Cycles Completed is affected by a few of the common losses:

- **Speed loss**—Operations which take a long time to complete will result in fewer cycles.
- **Planned downtime and Breakdowns**—You will be able to see the effect of downtime on Cycles Completed, as it will be unexpectedly lower.
- **Minor stops and Faults**—These will reduce the Cycles Completed for a particular time period.
- **Inefficiency**—If the robot's trajectory has not been well optimized, the Cycle Time will increase, which reduces the Cycles Completed in any particular time period.
- **Wait Time**—Longer Wait Times will reduce Cycles Completed.

ROBOTIQ UNIVERSAL ROBOTS

How to Measure Cycles Completed

Cycles Completed is one of the simplest metrics to measure, but there are a few different ways you could do it:

1. The Imprecise Way
The traditional way to measure Cycles Completed is to count the number of processed objects that are stacked up at the end of the robotic cell. If your robot processes multiple products per cycle, you can use this information to calculate the theoretical number of cycles.

Although this is easy to do, it has a couple of disadvantages:

- **It doesn't actually measure Cycles Completed**—This is actually a measure of Yield. It assumes that the robot has processed the number of products that it should have in every cycle. However, suppose that the robot occasionally drops a product from its grasp without detecting it—you couldn't measure the cycles when the robot was holding nothing.
- **It requires human intervention or additional sensors**—You need some way of counting the products as they leave the robotic cell. A human operator could do this or you could integrate a part detection sensor. Either way, this requires extra work, either for manual logging or to integrate the sensor with the robot.

2. The Back-of-the-Napkin Way
You can estimate the Cycles Completed by using the robot's theoretical Cycle Time (as described in the previous section). Simply calculate it using this equation:

$$\text{Cycles Completed} = \frac{\text{Robot Shift Time}}{\text{Theoretical Cycle Time}}$$

This will give you a rough indication of how many products the robot can process. It is only theoretical so it cannot be used to detect problems during the Operate Phase, but it's a useful calculation during the Design Phase when you have no access to the robot.

3. The Time-Consuming Way
The most involved method of obtaining the Cycles Completed is to measure it manually. This gives a fairly accurate snapshot of the robot's capabilities but it takes time. Use the following steps:

1. Start a stopwatch and start the robot.
2. Every time the robot completes a cycle, count it. Make a note of any failed cycles.
3. When a certain time period passes (e.g.10 minutes) write down the number of Cycles Completed.
4. To calculate the Cycles Completed per hour, for example, multiply by 6 (if you measured for 10 minutes).

The major problem with this method is that it does not provide a way to continually monitor the Cycles Completed. It is accurate at the moment you calculate it, and may be useful during the Integrate Phase, but it is not a long term solution.

4. The Quick-and-Dirty Coding Way
Counting cycles is very easy in the robot code. You simply increment a variable every time a cycle is completed. The method has similar issues to the Cycle Time method, in that you you either have to program the robot to store the data to the controller or send it across a network. However, there are usually fewer data points with Cycles Completed so this is less of an issue.

5. The Easy, Accurate Way
The final way to measure Cycle Time is to use cobot-specific logging software, such as **Robotiq Insights**. This communicates directly with your collaborative robot to count the Cycles Completed as they happen. It tells you, in real-time, how many cycles the robot achieves and allows you to quickly detect any issues.

Insights also gives you the ability to make custom events and count these as well as (or instead of) Cycles Completed. For example, if your robot processes multiple objects per cycle, you can set up Insights to trigger an event every time an object is processed. This will give you a more accurate measure of Yield. The software allows you to easily change the name "Cycles Completed" to something which better matches your application.

HOW TO CALCULATE YIELD WITH CYCLES COMPLETED
The Yield is an important metric in manufacturing because it determines how many products are processed compared how much waste is produced. If you already calculate the Yield for your business, you may want to know how you can combine the Cycles Completed KPI with your existing calculations. There are at least three different types of Yield commonly used in business, some of which can use Cycles Completed as a basis for their calculation:

1. Final Yield
This is the Yield for the entire process. It is calculated by combining all the individual Yields from the stages of the process, as described below. It does not take into account any re-workings. The term Unit here can be used to describe whatever it is that you are processing with your robot. One unit could refer to, for example, one part produced, one inspected product, one pick-and-placed book, etc.

The Final Yield is calculated by taking the total output and dividing by the total input:

$$\text{Final Yield} = \frac{\text{Total Units Out of Whole Process}}{\text{Total Units Into Whole Process}}$$

2. First Pass Yield
This the Yield for an individual step of the process. For example, you can calculate the First Pass Yield of your robotic cell.

The most accurate way to calculate First Pass Yield is to manually count the parts that go in and come out of each process. This is the traditional way to do this in manufacturing. You would then use this equation:

$$\text{First Pass Yield} = \frac{\text{Number Units Out}}{\text{Number Units In}}$$

However, this process can take a lot of time. The Cycles Completed KPI can be used to easily calculate the First Pass Yield for your robotic process using this equation:

$$\text{First Pass Yield} = \frac{\text{Number Good Units}}{\text{Cycles Completed}}$$

To get an accurate Number of Good Units, you will either have to count up the completed units manually or use some sort of automatic inspection process. However, you can also estimate the number of good units by counting the number of failed cycles (e.g. when the robot detects that a part has been dropped or misplaced, or a fault occurs).

$$\text{Estimated Good Units} = \text{Cycles Completed} - \text{Failed Cycles}$$

The First Pass Yield calculation assumes that there were no re-workings in the process. This is usually a reasonable assumption, since re-workings require human input. Robots operate autonomously.

3. Rolled Throughput Yield

This is the probability that all steps in the process will produce a unit with no defects. It is a cumulative measure of how many defects there are in the previous steps, including re-workings, and is more realistic than the other methods. It is calculated by subtracting the number of re-workings that were necessary at each stage.

$$\text{Rolled Throughput Yield per Stage} = \frac{(\text{Number Units Out} - \text{Scrap} - \text{Reworks})}{\text{Number Units In}}$$

If you have any re-workings, you can calculate them using the Cycles Completed KPI, as there will be more Cycles Completed than units put into the cell. This value will be zero if there were no re-workings:

$$\text{Reworks} = \text{Cycles Completed} - \text{Number Units In}$$

Worked Example

You can then use these equations to calculate the three yields.

Using the basic Final Yield equation above, you would get the following value:

$$\text{Final Yield} = \frac{\text{Total Units Out}}{\text{Total Units In}} = \frac{21}{30} = 70\%$$

The table below shows the yield values obtained by the three different methods:

Process	Type	Input Units	Output Units	Scrap & Reworks	First Pass Yield	Rolled Throughput Yield
1	(non-robotic)	30	26	4 scrap 2 rework	26/30 = 86.7%	(26-4-2) / 30 = 66.7%
2	(non-robotic)	26	22	4 scrap 1 rework	22/26 = 84.6%	(22-4-1) / 26 x 66.7% = 43.6%
3	(non-robotic)	22	22	0 scrap 0 rework	22/22 = 100%	22 / 22 x 43.6% = 43.6%
4	(non-robotic)	22	21	1 scrap 0 rework	21/22 = 95.4%	(21-1) / 22 x 43.6% = 39.6%

These values lead to a Final Rolled Throughput Yield of:

$$\text{Final Rolled Throughput Yield} = 39.6\%$$

Clearly, the basic Final Yield above gave an over-optimistic value for the Yield of the whole process.

Each type of Yield has its advantages. The advantage of calculating First Pass Yield for each stage is that you can clearly see which stages in the process are causing problems.The Rolled Throughput Yield then shows the effect of these issues on the overall process. You can use this information to improve the overall productivity of the process.

KPI 3. Utilization

Utilization measures how long a robot is being used compared to how long it could, theoretically, be used. After all, your robot is only useful when you actually apply it to tasks within the business.You won't get a ROI if it's just sitting there gathering dust in a corner.

We define Utilization as:

$$\text{Utilization} = \frac{\text{Time Robot Runs a Program}}{\text{Total Time}}$$

For example, if the robot was running for 15 minutes between 3 pm and 4 pm, the Utilization would be 25% (15 min/60 min).

You may be wondering: why would anyone ever leave a robot unused?

There are several reasons (which we'll discuss in a moment) but often the causes of consistently low Utilization are lack of planning and/or understanding of a robot's capabilities. Utilization provides a way to measure where this lack of planning is affecting the operation of the robot.

HOW UTILIZATION APPLIES TO COBOTS

There are several ways of measuring Utilization in a businesses. The most often used ways come from manufacturing. Some of these methods factor in the Yield, Efficiency, and other properties. However, many of the methods are not ideal for evaluating collaborative robots because they depend on other processes within the business.

The most useful way to measure cobot Utilization is to detect the percentage of time when it is running a program. It can be assumed that a robot which is running a program is actually performing a task. Of course, this does not say anything about the robot's efficiency at the task — you need the Efficiency metric for that— but it ensures that the robot is not sitting idle.

One of the things that makes collaborative robots unique is the ease with which they can be switched between tasks. So, we might use the same robot for a CNC Tending task, then move it on to a Packaging task, then a Polishing task, etc. The robots are easy to move around the factory floor and it's quick to launch a new program. Even if you only run each task for a couple of hours, you could still achieve high Utilization of the robot.

6 FACTORS THAT AFFECT UTILIZATION

1. **Planned Downtime**—Time spent on planned maintenance to a cell, for example, will affect the Utilization of the robot.
2. **Unplanned Downtime**—Small maintenance stops, mechanical problems and anything which causes unplanned downtime of the robot can affect its Utilization.
3. **Switch Time**—The time it takes to move the cobot between tasks will affect its Utilization, as it won't be running a program while it is being moved.
4. **Scheduling Losses**—Utilization will go down when the robot is sitting around doing nothing. You can reduce the effect of this by rescheduling the process to use the robot more.
5. **Breaks**—Small breaks, like lunch breaks or training, may come under the previous category of "planned downtime." However, it is important to highlight breaks separately because collaborative robots can run without supervision. Since it's unnecessary to turn them of when workers go on a break, you can increase Utilization by scheduling operations during breaks.
6. **Robot Training**—A major factor that affects robot Utilization is the knowledge level of its users. If workers are untrained in using the robot, they may not see ways to improve its Utilization. Even a small amount of on-the-job robotics training can prompt workers to think of new ways they could use the robot.

COMMON LOSSES WHICH AFFECT UTILIZATION

Utilization is mostly affected by three of the common losses:

- **Planned and unplanned downtime**—Any interruption of production which requires operators to stop the robot's program will present itself as a low value of Utilization.
- **Minor Stops**—Whenever the robot is in another state (idle, emergency stop, protective stop, safeguard stop, or initialization) this will lead to lower Utilization.
- **Lack of use**—A major cause of low Utilization is when the robot sits idle when it could be being used for another task.

HOW TO MEASURE UTILIZATION

Utilization can be a tricky metric to measure manually. Nothing is happening when the robot is sitting idle so it's hard to measure the exact moment when it finished its program. It's best to have it logged automatically in the robot's software, if possible.

1. The Imprecise Way

One quick way to measure Utilization is to record the time when you start the robot on a new task. This will give you a very rough breakdown of the Utilization for each task. However, it does not tell you how long the robot was idle for.

For example, imagine you use the robot for three tasks over a four hour period, logging the following times:

- **Task 1** (Machine Tending)—Start @ 08:00 (Program Ended @ 08:47)
- **Task 2** (Palletizing)—Start @ 09:20 (Program Ended @ 10:50)
- **Task 3** (Unboxing)—Start @ 11:00 (Program Ended @ 11:30) EndTime @ 12:00

By only logging the start times, you would achieve an overall Utilization of 100%. This would not include the program end time because you would not have logged this information.

The start times alone would give you the following breakdown:

- **MachineTending** = 09:20-08:00 / 4 hours = 33.3%
- **Palletizing** = 11:00-09:20 / 4 hours = 41.7%
- **Unboxing** = 12:00-11:00 /4 = 25%

But this would be innaccurate.The real Utilization must take the time that the program ended into account, not just the time that the robot was moved to another task.

Here's what the real breakdown should be:

- **Machine Tending** = 08:47-08:00 / 4 hours = 19.6%
- **Palletizing** = 10:50-09:20 / 4 hours = 37.5%
- **Unboxing** = 11:30-11:00 / 4 hours = 12.5%

This gives a more accurate Utilization of only 69.6%, meaning that the robot sat unused for over an hour (30.4%).

2. The Back-of-the-Napkin Way
You can estimate the Utilization of a robot by taking the time you plan to use the robot and dividing it by the entire time period:

$$\text{Utilization} = \frac{\text{Planned Use Time}}{\text{All Time}}$$

To estimate the Planned Use Time, simply take your "Cycle Time per unit" and multiply by the number of products that you want to process.

3. The Time-Consuming Way
A more accurate —but very time-consuming — way to calculate Utilization is to manually log the start and end times of the robot's program. You are unlikely to ever need to do this as robot controllers usually log the runtime of a program for you out-of-the-box. If, for some reason, your robot doesn't log the program run time, you could use a stopwatch to measure how long the robot takes to complete its task. You could do this while measuring the Cycle Time (see previous section).

4. The Quick-and-Dirty Coding Way
Robot controllers often output the runtime of a program automatically once it finishes. This is a very quick way to achieve a rough measure of Utilization. If your robot logs the program run time (or end time) on the screen of its teach pendant, you can simply note this down and use it to calculate Utilization according to the equation above.

The main issue with this method is that it requires extra work to log the data for later analysis. This is particularly problematic for Utilization, which is most helpful when you can compare long-term performance. You can either log the data manually, through File I/O or by sending the data over a network. However, like the problems outlined in the Cycle Time section, these add a lot of extra work to the coding.

5. The Easy, Accurate Way
The simplest way to measure Utilization is to use cobot-specific monitoring software, such as **Robotiq Insights**. This communicates directly with your collaborative robot and displays the Utilization of the robot in real-time. It shows detailed Utilization for each task individually and also provides the overall Utilization of the robot.

You can also set up **Insights** to automatically send you a SMS notification when the robot sits idle for a period of time, which allows you to increase Utilization even further.

WHY HIGH UTILIZATION ISN'T ALWAYS BETTER

Higher utilization is better, right?

Well, no. Not always.

There are some occasions where high utilization of your robot can actually cause lower productivity of your manufacturing process. This could happen when the robot is programmed inefficiently or when it affects the flow of the overall process.

High Utilization but Low Throughput

Consider these three scenarios:

1. A robot takes two hours to assemble 50 units (Utilization = 100%).
2. A robot takes 30 minutes to assemble 50 units, then stops the program and sits idle for 1.5 hours (Utilization = 25%).
3. A robot takes 30 minutes to assemble 50 units, then keeps assembling units for 1.5 hours even though they pile up at the next stage of the process (Utilization = 100%).

Which is best?

The second scenario is, even though it has by far the lowest utilization. Compared to the first scenario, the robot is completing the same task in a quarter of the time. And in the third scenario, the utilization is high but the robot is overproducing, resulting in piles of unnecessary Work in Progress.

Higher utilization is not better when it adversely affects the Throughput of the overall process.

Why does the robot in the first scenario take longer? It's likely that the robot has been programmed badly or is in a "reduced speed" mode when it does not have to be. In these cases, the Cycle Time metric would be much higher for the first scenario.

Do You Prioritize Utilization At All Costs?

Many manufacturing businesses prioritize Utilization over Throughput. This is a reflection of how business owners thought during the industrial revolution when machines were expensive. Factory owners wanted to run machines constantly to ensure they were getting their money's worth. However, this mindset leads to inefficient process planning, with Work in Progress piling up all over the shop floor.

Prioritizing the utilization of the robot over everything else can cause overall Throughput to suffer. It is much better to try to optimize the global throughput of your process. Utilization is a great KPI to help you improve the productivity of your robotic cell, if necessary. However, maximizing Utilization should not be your ultimate goal.

KPI 4. Efficiency

Efficiency goes hand-in-hand with Utilization. It defines the percentage of time that the robot performs productive work while running a program. We define it as:

$$\text{Efficiency} = \frac{\Sigma \text{ Robot Motion Times}}{\text{Total Time}}$$

For example, if a robot was moving for a total of 48 minutes between 5 pm and 6 pm its Efficiency would be 80% (48 min / 60 min).

The use of the robot's motion time is a reasonable measure of a robot's Efficiency, but it is not completely accurate. Motion does not necessarily equal production — you could, for example, make the robot dance around not doing productive work and it would give a high Efficiency value. As a result, you should make sure to optimize your robot's trajectories to ensure it is moving only the distances that are absolutely necessary.

Maximizing Efficiency is the key to getting the most from your robot. You can achieve high Efficiency by minimizing the time that the robot has to wait.

HOW EFFICIENCY APPLIES TO COBOTS

At least two different concepts of Efficiency are commonly used in business:

- **Production Efficiency**—This measures how well you use your machines or processes.
- **Energy Efficiency**—This measures how much energy a machine uses, comparing the amount of energy put into the machine with the energy you get out of it.

For robotics, we are most interested in the production efficiency because we can control it. Energy efficiency is a property of the robot, so it is mostly the responsibility of the robot manufacturer. The simplest way to make sure your robot is energy efficient is to choose the right robot. Also, you should always turn it off when it is not being used for a long time.

The standard way to measure production efficiency comes from manufacturing businesses. It is to compare the actual output of a process (in terms of number of products) with its effective capacity. This indicates whether the production is running behind or ahead of target.

$$\text{Production Efficiency} = \frac{\text{Actual Output}}{\text{Effective Capacity}}$$

However, this is not the best way to measure the Efficiency of a collaborative robot. This definition could lead you to mistakenly think that the robot is running efficiently, while in fact you could still improve its operation.

5 Factors Which Affect Efficiency

1. **Waiting**—A major time loss for collaborative robots is waiting for other processes to finish. This is most often seen in applications like machine tending, where the robot waits for another process to finish before it continues
2. **Robot Training**—Operators who are untrained with the robot may not program it to follow the most efficient path. Even a little robotics training can empower workers to continually improve the way they use the robot.
3. **Complexity**—Complex tasks and programs are more likely to cause the robot to enter faults. You can reduce the effect of these faults with thorough testing.
4. **Part Placement**—Consistent part placement is always helpful when using robots. Although robots with sensors can handle irregularly placed objects, unintended wait times are more likely to occur.
5. **Setup times**—There are often apparent losses of Efficiency at the start of a new task, when the robot has either been moved from a different task or has been switched off for a while.

This is because the effective capacity in the equation is based on a theoretical measure of capacity of the entire manufacturing cell—it doesn't pinpoint the efficiency of the robot itself.

COMMON LOSSES WHICH AFFECT EFFICIENCY

Efficiency is affected by several of the common losses.

- **Wait Time**—We have found that Wait Time is the biggest cause of cobot inefficiency for most tasks. This can occur, for example, when the robot is waiting for other processes to finish before it can continue.
- **Unplanned downtime**—Small breakdowns and other interruptions which do not require the robot program to be stopped will be visible as low values of Efficiency.
- **Minor stops**—Misalignment or dropping of objects can often be solved without stopping the robot. This may show up in the Efficiency value if the robot has been programmed to wait for the object to be replaced. This is really a case of ineffective programming, since the robot should detect an error if the object is missing.
- **Inefficiency**—Programming that's been carried out by less skilled operators may be less efficient, such as if they direct the robot to wait unnecessarily.

How to Measure Efficiency

Efficiency can be quite simple to calculate, depending on the resources you have available to you.

1. The Imprecise Way

Many manufacturers measure "Machine Efficiency" on the shop floor. This is a simple measure of machine runtime.
This is certainly a quick calculation to make (you simply measure the time that the machine was switched on then divide this by the total shift time). However, it is not a good measure of robot Efficiency.

In fact, this definition has more in common with robot Utilization than it does with Efficiency. It won't indicate how well the robot is performing or how you could improve its program.

2. The Back-of-the-Napkin Way

One way to estimate the Efficiency is to take the Planned Use Time for the robot (as described above in the section on Utilization) and make an educated guess about the Wait Time.

$$\text{Efficiency} = 100\% - \frac{\text{Wait Time}}{\text{Planned Use Time}}$$

You can estimate Wait Time by summing the amount of time that other processes take to deliver product to the robot. See the previous section on Cycle Time for an example of this with a machine-tending robot.

This is an inaccurate way of measuring Efficiency, so it's only useful during the Design Phase. However, it can provide a rough benchmark with which to compare the real Efficiency once you have measured it.

3. The Time-Consuming Way

The manual method for measuring efficiency involves measuring Wait Time with a stopwatch—see the next section for instructions on how to do this. You can then use the equation shown in The Back-of-the-Napkin Way to calculate efficiency.

4. The Quick-and-Dirty Coding Way

You can program the robot to calculate the Efficiency of a particular task quite easily. First, keep a running total of all the Wait Times in a variable, as described in the next section. You should also keep a running total of the total time in a second variable.

Whenever you need to access the Efficiency value (e.g. when the program terminates) use the equation provided above.

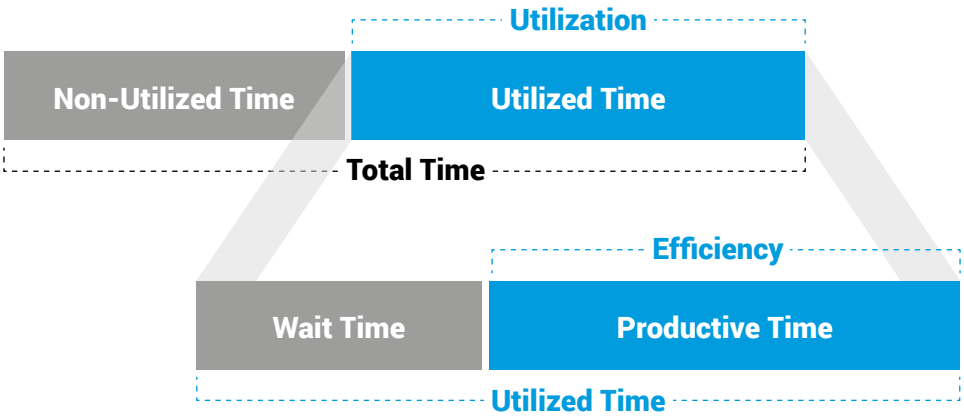
This method has the same issues that have been outlined in the previous sections: i.e., it is a useful method for calculating Efficiency quickly during the Integrate Phase, but it is not a good long-term solution.

5. The Easy, Accurate Way
The simplest way to measure Efficiency is to use cobot-specific monitoring software, such as **Robotiq Insights**. This communicates directly with your collaborative robot in real-time, and provides Efficiency values both for a specific task and for the robot' overall performance.

Utilization vs. Efficiency
What's the difference between Utilization and Efficiency?

Utilization measures the percentage of "Total Time" (e.g. one hour, one day, etc.) that a robot is running a program. Non-Utilized Time occurs when the robot is Idle, in a Stop state or Initializing. Efficiency measures the percentage of Total Time that the robot is actually moving.

The Wait Time and Efficiency should both add up to the Utilization. For example, if Wait Time is 5% and Efficiency is 40%, Utilization will be 45%.



When should you use each one?

Utilization is good for:

- **Finding a robot for a new application**—If you have a robot with low Utilization, it could be used for other applications in your business. For instance, you might be able to move the existing robot to a new task during its downtime instead of purchasing a new robot for the job.
- **Improving production scheduling**—Utilization can be a helpful metric when you're planning your production flow. Identify stages of the process that have high Utilization but low Throughput to figure out where the flow can be improved.
- **Estimating and minimizing switch times**—There's almost always a loss of Utilization when you move the robot from one task to another. Very low Utilization between tasks can indicate where these switch times could be optimized.

Efficiency is good for:

- **Identifying Inefficient Programs** — Consistently low values of Efficiency for a particular task may indicate that the program for that task needs to be better optimized. Some tasks, like machine tending, often have higher Wait Times.

- **Improving flow of a cell**—Low Efficiency combined with high Wait Time indicates that the cell could make better use of the robot. If you find that the robot is waiting a long time for a particular process to finish, for example, consider whether it could perform another task while it waits.
- **Identifying improvement points**—For some tasks, you may find that Efficiency remains constant, but the Cycles Completed and Cycle Time metrics change. Identifying why the Cycle Time is different during those cycles can help you improve the productivity of the robotic cell.

KPI 5. Wait Time

Wait Time is the percentage of time that the robot is waiting (i.e. not performing productive work) while it is running a program.

We define it as as a summation of all individual wait times:

Wait Time = ∑ Robot Static Times

where Robot Static Times are those durations in a robot's program where a program is running but the robot is not moving. It is an important metric for collaborative robots because time losses often occur when the robot is waiting for another process to finish. Here at Robotiq, we have seen many situations where the Wait Time is the main cause of inefficiency.

HOW WAIT TIME APPLIES TO COBOTS

Traditionally, Wait Time is defined as all non-processing time for a product—i.e. when it's undergoing neither value-added operations nor non-value-added operations. However, this definition is not directly applicable to robots.

Cobot Wait Time has a specific definition. It measures those times in the robot's programming when the program "hangs" while waiting for an external signal.

Some examples of external signals are:

- **Sensor Signals**—When the robot is using a part-detection sensor or Robot Vision, for example, it will wait until an object appears before moving to grasp it.
- **Communication Messages**—When the robot has been integrated with another machine (e.g., a CNC machine) it will wait for a message from the other controller to indicate that its process is finished.
- **Human Input**—When the robot controller has been programmed to wait for an operator to enter information or confirm an action, it will wait for this signal before it continues with its program. This is less common in manufacturing applications, but it can sometimes arise.

COMMON LOSSES WHICH AFFECT WAIT TIME

Unlike the other KPIs in this list, Wait Time is itself a "common loss," so it's less affected by the other common losses (e.g., breakdowns, speed loss, etc.). It's an important metric for collaborative robots because it highlights an inefficiency which is hard to spot using the other KPIs alone.

HOW TO MEASURE WAIT TIME

There are several different ways to calculate the robot's Wait Time.

1. The Imprecise Way

One calculation for Wait Time that you may have encountered before is often used in machine shops, where Wait Time is calculated for each human operator individually. It's usually incorporated into the Process Time for that operator, which combines the work time, travel times and wait times. However, this does not reflect the use of the machine. It is only a way of measuring the operator's productivity.

This method is not directly applicable to collaborative robots — first, because no human operator is present in a robot process. And second, because this definition of Wait Time is just an umbrella term for any non-work time that cannot be classified as traveling. As we've discussed, robot Wait Time has a specific definition.

2. The Back-of-the-Napkin Way
A quick way to estimate the Wait Time is to add up all the Cycle Times for the processes that the robot will be waiting for. For example, if the robot will be machine-tending a single CNC machine, the Wait Time will be approximately the same as the processing time of the machine.

This is only a rough measure, as it does not account for any small waiting times which may add to the overall Wait Time. Also, it doesn't account for any parts of the program that the robot carries out while the machine is operating, which may reduce Wait Time.

3. The Time-Consuming Way
The manual method to measure Wait Time is to get out your stopwatch. If you already measured your process to obtain the Cycle Time (see previous section) then you may already have this information. If not, use the following process:

1. Set up the robotic cell that you want to measure.
2. Define the sequence of events for the whole task, as shown in the previous Cycle Time section.
3. Start the process. Every time the robot stops, while waiting for another stage of the process to finish, start the stopwatch and record the time.
4. Let the robot go through the sequence several times — the more the better.
5. Calculate the average Wait Time for each stage of the process.
6. Finally, add all of these times to give the overall Wait Time for the process.

This is a more accurate way to measure Wait Time than the previous two methods. However, it is unnecessarily time-consuming as it is fairly easy to record Wait Time within the robot's program. As a result, you probably won't do it very often, which makes it hard to catch new, unexpected Wait Times as soon as they appear.

4.The Quick-and-Dirty Coding Way
You can program the robot to record the Wait Time by running a "timer" in your code whenever the robot is static. For example, the robot could be waiting for an external signal to be triggered or an object to appear in front of a sensor. You would run the timer until the signal was triggered.

One way to program such a timer is by incrementing a variable for every loop of the program, then using this to calculate the time based on the controller clock rate. For example, say your robot controller runs at 125Hz. Each cycle, you would increment the count variable by one. When the external signal triggers, the variable has counted 240. You then multiply this by 0.008 (1/125) to get a Wait Time of 1.92 seconds. You can then add this to a variable containing a running total of the Wait Time for the whole program.

Although this is a reasonably simple code to implement, it can get complex if there are many different causes of Wait Time in your program. You also run into the same problems outlined in previous sections: i.e., it is a useful method for calculating Efficiency quickly during the Integrate Phase, but it is not a good long-term solution.

5. The Easy, Accurate Way
The simplest way to measure Wait Time is to use cobot-specific monitoring software, such as **Robotiq Insights**.This communicates directly with your collaborative robot and displays the Wait Time of the robot in real-time, allowing you to immediately tell when the robot is waiting for something.

6 CAUSES OF COBOT WAIT TIME (AND SOLUTIONS)
Waiting usually occurs when the robot has to rely on other processes, or human operators, to complete their allocated tasks before it can operate.

For example:

1. **Waiting for other processes to finish**—We see the highest Wait Times in CNC machine-tending applications, where the robot has to wait until the machining operation finishes to continue with its program. This can be reduced by using the robot to tend multiple machines or giving it extra tasks to do while it's waiting.
2. **Waiting for product to arrive**—Some tasks require the robot to wait for a product to arrive before it can carry out its task. Boxing and palletizing tasks, for example, may cause some delays at the end of a process. This can be reduced by introducing a small buffer (i.e., a pile of product) before starting the robot's task.
3. **Waiting for product to be removed**—A similar issue is waiting for product to be removed before the robot can continue with its task. For example, a boxing robot may have to wait for the full box to be removed before it can start filling up the next one. This can be reduced by having finished products automatically moved away from the robot.
4. **Transportation times**—The transportation of Work in Progress between different stages of the process can cause the robot to wait.This can be reduced by shortening the transportation distances or preemptively transporting the product before the robot has finished its current batch.
5. **Missing steps of the process**—Some tasks involve a robot collaborating with human operators. This can lead to Wait Time if the operator has to leave, which can be reduced by providing the robot with enough of a buffer to account for the time that the operator will be gone.
6. **Breakdowns**—When the robot itself breaks down, this will be represented in its Utilization KPI. However, machine breakdowns elsewhere in the process may present themselves as Wait Time for the robot.This can be reduced by moving the robot to another task until the breakdown is fixed.

(Cobot Metric 6: Disconnected)

Another metric you may want to measure is the Disconnected time. It's not a Key Performance Indicator, but it is an increasingly important metric for network-connected devices. After all, network connection is the backbone of the Internet of Things (IoT), and a network-monitored cobot is basically functioning as an IoT device.

Disconnected indicates the percentage of time during which the robot was disconnected from the network because it has lost connection from the network.

Here's what Disconnected means:

- **Disconnect = 100%**—This means that the robot was disconnected from the network for the entire time period.
- **Disconnect = 0%**—This means that the robot was connected to the network for the entire time period.
- **Any other value**—This means that the robot was disconnected from the network for a portion of the time period. For example, if you are measuring one hour and Disconnect = 40%, it was disconnected for 24 minutes during the hour.

HOW DISCONNECT TIME APPLIES TO COBOTS

Collaborative robots can operate even when they are not attached to a reliable network. However, connection is necessary to receive real-time information about the robot's operation.

For example, imagine you have a depalletizing task and you are looking around the workshop for a spare robot to apply to the task. The quickest way to find a suitable robot is to log on to your cobot logging software and look for a robot that has 0% Utilization at the moment. In this case, you will need a reliable network connection to receive information from all of the robots on the shop floor.

Sometimes, your network connection may be unreliable or choppy. In this case, it's helpful to have robots that can store their KPIs internally while the network isn't present. **Robotiq Insights**, for example, can store KPIs for up to 30 minutes without a network connection. Once the connection is reestablished, the robot will send all its stored information to the main system.

Disconnect as a Measure of Network Quality

The Disconnected metric can also be a useful way to judge the quality of your network, allowing you to quickly spot when there are problems. When coupled with Insight's SMS notifications, it can even function as an early warning system of network trouble when you are away from the building.

You will typically see these effects:

- **Disconnect = 100%**—This could indicate a complete loss of network connection, especially when all robots have this reading. Since it's indistinguishable from the case when the robot was switched off, you may need to investigate further.
- **Disconnect value changes between time periods but other KPIs are relatively normal**—This could indicate a bad network. The Disconnect value indicates a loss of connection, but the other KPIs suggest that the robot is still operating even when it is disconnected from the network, as the robot will buffer KPIs for up to 30 minutes when no connection is present.

6 Steps for Using KPIs Effectively

Picking your KPIs is only the start. Once you are reliably logging them, you will want to monitor them and use the data to improve the performance of your collaborative robotic cell.

Here are six steps for using cobot KPIs effectively.

1. ALIGN THE KPIs WITH BUSINESS GOALS

You should be able to link all your KPIs to the goals of the company. This ensures that you're measuring data that contribute to the continuing improvement of the business. Although this might sound like a lot of work, it's actually quite simple—you probably do it already.

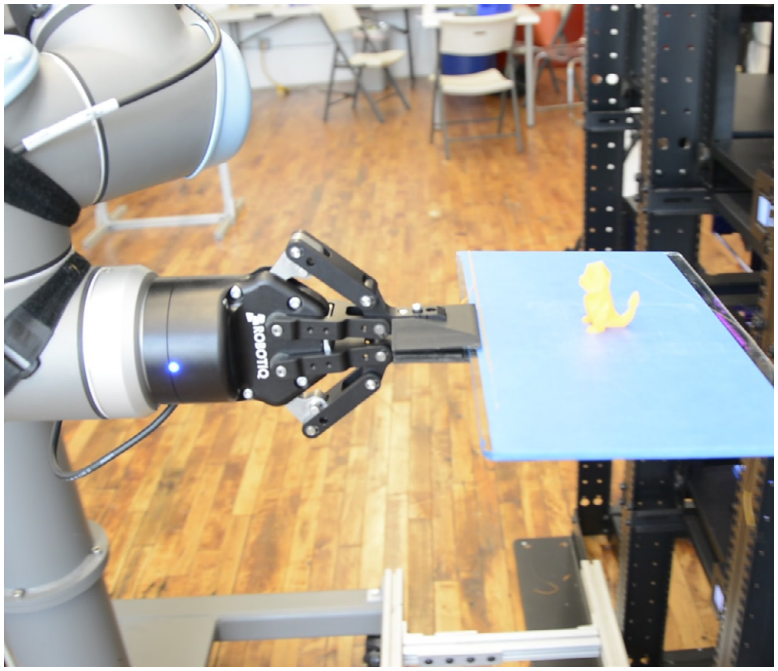
Think about what the business is trying to achieve right now. You might know this from meetings or recent company-wide presentations.

Is it trying to atract more customers? Increase production? Improve quality?

For each goal, consider how you can use the collaborative robot to get closer to the desired result, even if it's only in a small way. Determine which KPIs are going to be most important for achieving those results.

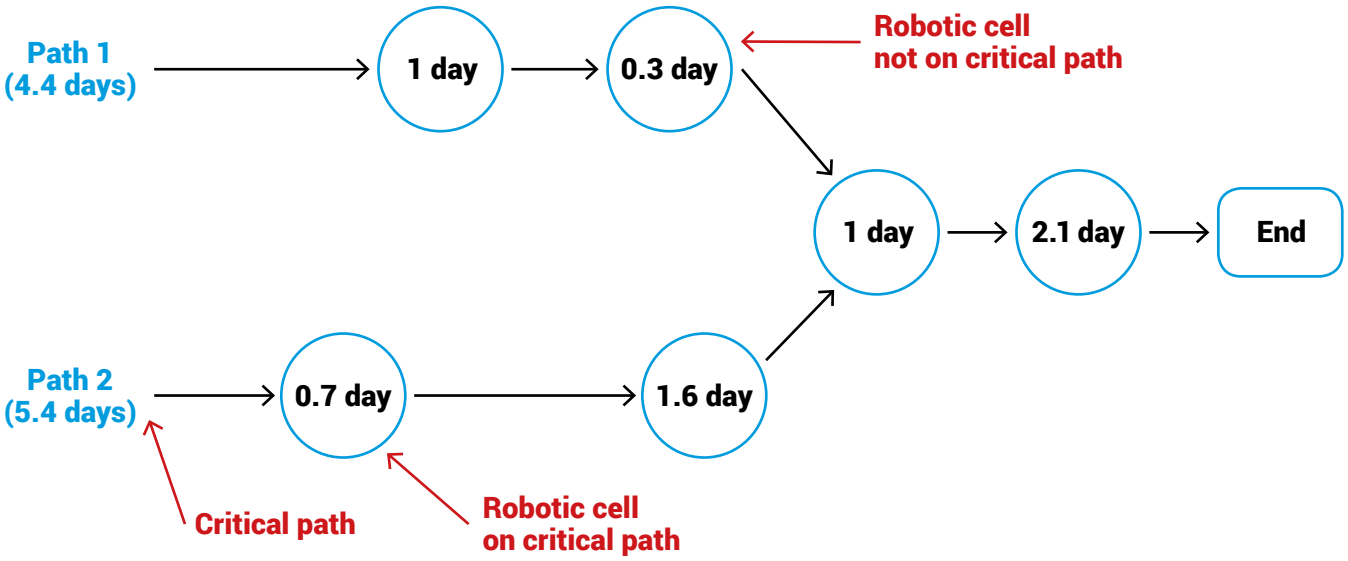
2. FIND OUT IF THE ROBOT IS ON THE CRITICAL PATH

The critical path is the sequence of operations that determines the minimum time it takes for your product to pass through the entire process. If the robotic cell is not on your critical path, it will not affect this minimum time and it may make more sense to focus on improving the performance of another step in the process.



To work out if your robot is on the critical path:

1. Make a visual representation of all the operations in your process, including the various paths that the product takes through the process and the time it takes to pass through each stage.
2. Sum up all of the times for each path to give an overall time for the process.
3. The path with the longest time is your critical path. It determines the minimum time it takes for the product to pass through the process.



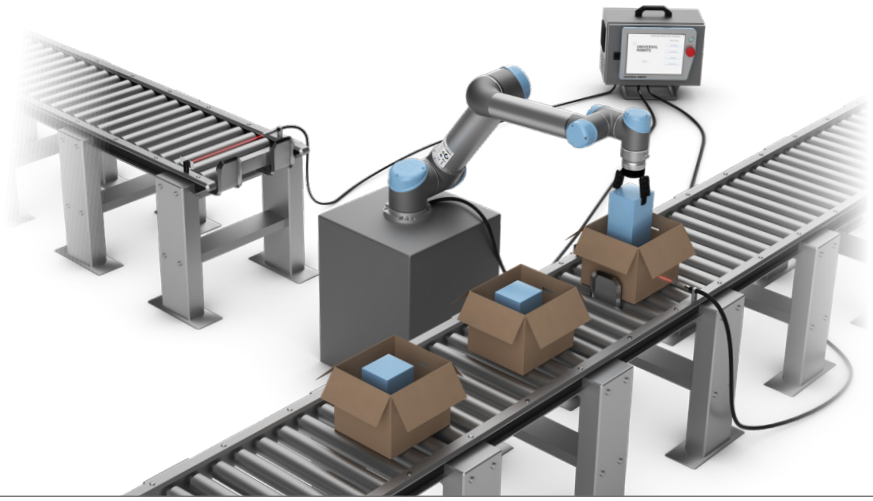
If your robot is part of the critical path, it makes sense to focus on optimizing its performance. If your robot is not on the critical path, your efforts may be better spent optimizing other stages in the process. Consider how you could add a collaborative robot to a stage on the critical path to improve the overall process time.

3. FIND OUT IF THE ROBOT IS A BOTTLENECK

If the robotic cell itself is a bottleneck, you will definitely want to use KPIs to improve its performance. Signs that the robot may be a bottleneck include:

- Work in Progress is piling up at the stage before the robotic cell.
- The robot has a long Cycle Time and/or low Cycles Completed.
- The robot is running at full Utilization but it still does not meet the Throughput targets.
- If the robot's Wait Time is high, you may find that other processes within the robotic cell (such as a CNC machine that the robot is tending) are causing the bottleneck.

If you find that the robotic cell is the bottleneck, use the KPIs to identify how you could improve its function.



4. USE THE ROBOT KPIs TO IDENTIFY ANY PROBLEMS

Monitor the KPIs over time and compare the cobot's performance for different tasks and shifts. This way, you'll catch potential problems before they start affecting other stages of the process.

Look at each of the five KPIs and ask yourself:

- Which KPIs are showing lower values than expected?
- Which low values can't be easily explained?
- Which of the common losses are causing the low values?
- What is causing any high Wait Times and/or low Efficiency values?
- How could the Utilization of the robot be improved?

5. INVESTIGATE AND RESOLVE THE CAUSES OF DELAY

Once you've identified any losses, go to the robotic cell and investigate the causes. Remove or reduce the losses wherever possible, and then use KPIs to monitor the effects of your changes.

6. REPEAT FOR CONTINUOUS IMPROVEMENT

Optimizing the performance of your robotic cell is a continuous process. Whenever you have made any changes to the robotic cell, keep an eye on the KPIs to see what effect those changes have in the long term. Make sure to monitor your KPIs over time and investigate any differences between times, teams and tasks.

When you have improved your first cobot application as much as you can, consider how you could add collaborative robots to other stages in the process to boost overall productivity. Use your KPIs as a way to compare the robot's performance between the different tasks and strive for continuous improvement.

About Robotiq

Robotiq's Lean Robotics methodology and products enable manufacturers to deploy productive robotic cells across their factory.

They leverage the Lean Robotics methodology for faster time to production and increased productivity from their robots. Production engineers standardize on Robotiq's Plug + Play Components for their ease of programming, built-in integration, and adaptability to many processes. They rely on Flow's software suite to accelerate robot projects and optimize robot performance once in production. Robotiq is the humans behind the robots: an employee-owned business with a passionate team and an international partner network.

Let's Keep in Touch
For any questions concerning robotic and automated handling or if you want to learn more about the advantages of using flexible electric handling tools, contact us: **www.robotiq.com**

Follow Robotiq:

About Universal Robots

Universal Robots is a result of years of intensive research in robotics. The six-axis robot arms weigh as little as 40 lbs. with reach capabilities of up to 51 inches. Repeatability of +/- .004" allows quick precision handling of even microscopically small parts. After initial risk assessment, the collaborative Universal Robots can operate alongside human operators without safety guarding. If the robots come into contact with an employee, the built-in force control limits the forces at contact, adhering to the current safety requirements on force and torque limitations.

Universal Robots was co-founded in 2005 by the company's CTO, Esben Østergaard, who wanted to make robot technology accessible to all by developing small, user-friendly, reasonably priced, flexible, industrial collaborative robots. Since the first collaborative robot (cobot) was launched in 2008, the company has experienced considerable growth with the user-friendly cobot now sold in more than 50 countries worldwide. The company, which is a part of Teradyne Inc., is headquartered in Odense, Denmark, and has subsidiaries and regional offices in the USA, Spain, Germany, Italy, Czech Republic, China, Singapore, India, Japan, Taiwan and South Korea. Contact us: **www.universal-robots.com**

Follow Universal Robots: