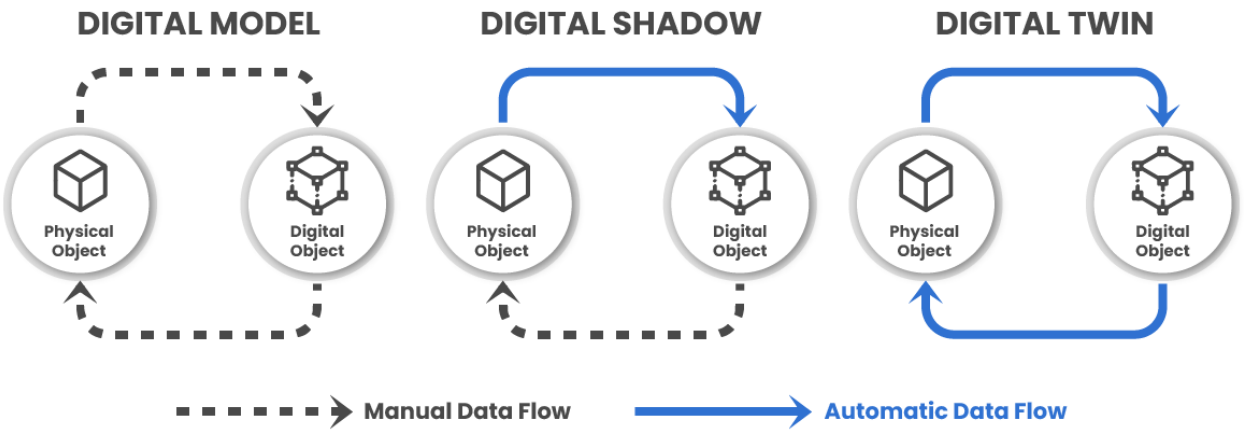
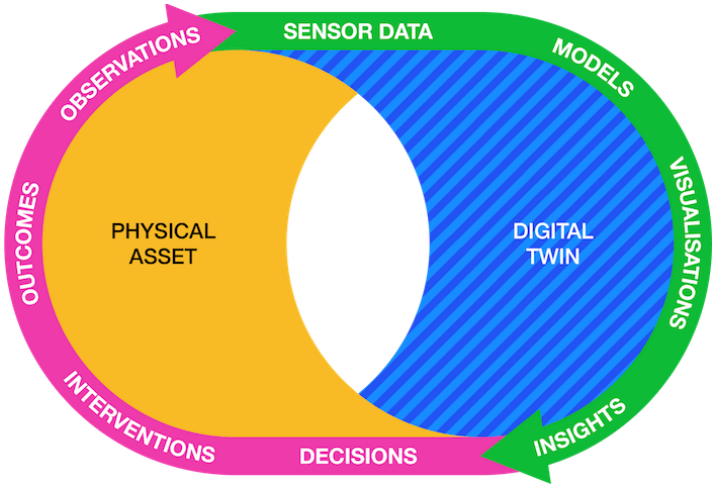
What is a digital twin?

A digital twin is a digital object that represents a real object or system in the physical world. Data is fed real-time into the digital twin and that data is used to create visualizations, provide insights, make decisions, and initiate human intervention in the real-world system.

A digital twin is more than just a digital model of something that exists in the real world. It is different because of the flow of data to and from the system.



What a digital twin does:



Intro: https://youtu.be/60eCpw0Toy4

What is a digital twin? Classroom Activity

Step 1: Write a program that turns an LED on when you push a simple button and turns it off when you release the button.

You're working in the factory's control room, and your first task is to monitor machine health. To start, you need to create a simple control system. You're tasked with wiring up a button that will trigger an LED light to turn on. This light will act as a status indicator for a key machine in the production line. When the button is pressed, the LED will illuminate to indicate that the machine is online and operational. This is a simple way to test your ability to send signals from the physical world (the button) to a digital system (the LED), which is a critical first step in implementing a digital twin system.

On your Raspberry Pi, a button is an input and the LED is an output. The button sends information to the Raspberry Pi every time the button is pressed or released. The Scratch program receives the input, runs a script you write, and the script sends an output to the LED to light it up. **Write a program that turns an LED on when you push a simple button and turns it off when you release the button.**

You will need:

- Raspberry Pi with a display connected and mouse and keyboards
- Scratch 3 software installed on your Raspberry Pi
- Breadboard
- 4 male-to-female jumper wires (you will make 4 connections from the Pi board to the breadboard)
- Simple LED
- Simple Button
- 220 Ohm resistor

Helpful link: Controlling an LED | Button switches, Scratch 3, and Raspberry Pi 4

Step 2: Make the LED light up when the temperature goes over 80 degrees Celsius and turn off when it goes below 80.

As the technician responsible for monitoring equipment, you need to ensure the machines don't overheat. A digital twin of one of the machines is tracking the temperature in real-

time. You'll set up the system so that if the temperature sensor detects that the machine temperature exceeds 80°C, the LED will light up, signaling an overheating issue. This action is crucial for preventing potential damage and ensuring the manufacturing process runs smoothly.

On Scratch on your Raspberry Pi, you can add extensions by clicking the "Add Extension" button ). The Raspberry Pi Sense Hat extension allows you to simulate changes in temperature, pressure and humidity, among other things. **Write a program that turns an LED on when you raise the temperature (on the emulator) above 80 degrees Celsius and turns it off when it goes below 80.**

You will need:

- Everything from Step 1
- Sense Hat Emulator program for the Pi
- Raspberry Pi Sense Hat extensions for Scratch

Step 3: Make a buzzer beep once per second when a key is pushed.

Now, let's add a step that simulates an emergency protocol. You've been asked to create an alert system that sounds off when a certain event happens. You'll wire up the buzzer and write a program that you can start and stop with the keyboard. This will serve as a test of the emergency alarm. **Write a program so that when the "Y" key on the keyboard is pressed, the buzzer starts beeping every second AND when the "N" button is pressed, the buzzer stops.**

In addition to what you have from steps 1 and 2, you will need:

- An active buzzer
- Two additional male-to-female jumper wires

Helpful link: https://projects.raspberrypi.org/en/projects/physical-computing-with-scratch/4

Step 4: Make the buzzer go off when the pressure goes over 1000 millibars (mbar).

Next, you are tasked with creating a pressure monitoring system. In the factory, one of the machines operates under high pressure, and it's crucial to make sure that pressure levels stay within safe parameters. If the pressure exceeds 1000 mbar, the buzzer will sound off to

alert the team to check for potential system failure or malfunction. This is a vital part of maintaining safety in the workplace and preventing equipment damage. **Write a program that makes the alarm from Step 3 sound when the pressure goes above 1000 mbar on the emulator.**

You will not need any additional resources for step 4.

Step 5a: Set up the speed sensor to detect "widgets."

In the production line, your role involves monitoring the output rate of manufactured products, known as widgets. A speed sensor needs to be installed to track how many widgets are being produced per minute. You'll calibrate the sensor to ensure it accurately detects every widget passing through the production line.

A speed sensor transmits a beam of light to a sensor. When the beam is block (the sensor stops receiving a signal), the pin outputs a signal. We will imagine our factory uses a speed sensor to count the widgets as they go by on a conveyor belt. **First, write a program that detects widgets by turning on an LED when the beam on the speed sensor is broken (representing a widget passing through the speed sensor).**

In addition to previous steps, you will need:

- Speed sensor module
- Three female-to-female jumper wires (same on both ends, which is different than previous steps)
- Something to break the beam (anything will work: an index card, coin, or a sticky note)

Helpful Link: 1.7 Piggy Bank — SunFounder Ulimate Raphael Kit for Raspberry Pi documentation

Step 5b: Create a program that counts widgets as they pass through the speed sensor.

*Now that your speed sensor **detects** widgets, you need to create a program that **counts** every time a widget passes through the sensor. The sensor's readings will feed into the digital twin, giving you live data on the manufacturing process.*

Your speed sensor sends a signal when the beam is broken. **Write a program that increases a count by one each time a widget is detected (beam is broken).** You program

should increase the count by 1 each time the signal goes from low to high (this represents a widget being counted on the conveyor belt).

You will not need any additional resources for this step.

Step 6: Create an alarm (buzzer) that will go off when more than 10 seconds have passed without a widget.

As part of your task to ensure the efficiency of the production line, you need to detect when there is a delay or stoppage in the process. If more than 10 seconds pass without a widget being counted by the speed sensor, you'll set up an alarm (buzzer) to go off. This will serve as a reminder to operators that something may be wrong with the machine, and they need to investigate the delay. This ensures that production continues smoothly and helps minimize downtime.

The buzzer should already be set up from step 3 and the speed sensor is set up from the previous step. You will need to create a timer that resets to zero each time a widget is detected. **Write a program that tracks how much time elapses between widgets and sets off the buzzer if too much time passes without a widget (10 seconds).**

You will not need any additional resources for this step.

Helpful link: https://wiingy.com/blog/how-to-make-a-timer-in-scratch/

Step 7: Create a list of the time elapsed between widgets.

You are asked to monitor the production rate and identify any potential delays. To help with this, you will record the time between each widget being produced. This list will help the team analyze trends in the production line speed and identify when and why slowdowns are occurring. You're essentially building a logbook for the production process that can be analyzed later for troubleshooting and optimization purposes.

You now have a way to detect and count widgets and an alarm that notifies the production team there is a slow down in production. Scratch has the capability to make a list of values. Using this feature, **create a list of "time elapsed" between each widget.** So, not only will you count how many widgets, but you will keep a record of how long each widget took to make.

You will not need any additional resources for this step.

Step 8: Calculate the average time elapsed between widgets with a program.

As a final step, your task is to evaluate the overall efficiency of the production line by calculating the average time it takes between each widget produced. With this data, the factory can assess how well the production system is performing, and you can use the insights to make adjustments to improve output. The calculation will also be part of the digital twin model, which allows real-time optimization based on past performance.

Finally, you will use your list to calculate the average time to create a widget. Remember, the average is the sum of all "elapsed time" values on your list divided by the number of items on the list (Scratch refers to this as "length of <name of your list>"). **Write a program that calculates the average time to create a widget.**

Closing:

Congratulations! By completing these steps, you've effectively set up and simulated a digital twin of a manufacturing environment. You've implemented sensors, alarms, and data logging systems that help monitor the production line in real-time. By using the Raspberry Pi, Scratch, and the Sense HAT emulator, you've been able to create a digital replica of the physical factory system. This allows engineers and operators to predict, optimize, and troubleshoot the production process before problems even occur. Your work is a perfect example of how technology can be used to improve efficiency and safety in manufacturing settings.

What is a digital twin? Check for Understanding

1. What is a digital twin, and how does it relate to physical objects?

A digital twin is a dynamic up-to-date digital replica of a physical object.

2. Describe how digital twins can be beneficial in the construction and maintenance of buildings.

Digital twins can help with planning, design, construction, operations, and maintenance by providing real-time updates.

3. Explain the significance of NASA's use of mirroring technology in the 1960s.

NASA's use of mirroring technology was significant as it helped replicate systems in space, notably during the Apollo 13 mission.

4. What are some of the industries that have adopted digital twin technology?

The manufacturing, architecture, engineering, and construction industries have adopted digital twin technology.

5. Discuss the differences between descriptive twins, informative twins, and predictive twins.

Descriptive twins provide visual replicas, informative twins add operational data, and predictive twins leverage data for insights.

Controlling an LED | Button switches, Scratch 3, and Raspberry Pi 4 | Scratch | Coding projects for kids and teens

Buzzer: https://projects.raspberrypi.org/en/projects/physical-computing-with-scratch/4

Timer: https://wiingy.com/blog/how-to-make-a-timer-in-scratch/