Programmering af Mobile Robotter

RB1-PMR – Module 6: Sensors and Robot Behaviors

Jes Hundevadt Jepsen | jegj@mmmi.sdu.dk
SDU Drone Center, MMMI, University of Southern Denmark, Odense, Denmark

Agenda

- Recap of last module

- Sensors
  - Sensor categories
    - Internal versus External
    - Local versus Global
    - Active versus Passive
  - Sensor types
    - Binary sensors
    - Analog sensors
    - Digital sensors

- Robot Behaviors
  - Pseudocode
  - State machines
  - Flowcharts
  - Simple control (using feedback)

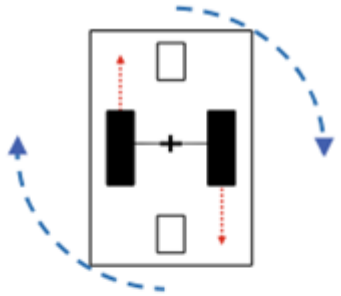
- Introduction to Extra Credit Activity #3

- Exercises

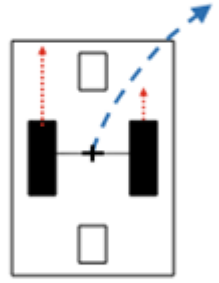
Recap

- Unmanned Systems (UAV, UGV, USV, UUV)
  - Levels of Autonomous Systems
- Unmanned Ground Vehicles (UGV)
  - Different types (Wheeled, Tracked, Legged)
  - Different drive controls (Wheeled)
    - Single Wheel, Ackermann, Omni-Directional, Differential
    - Differential drive control
      - Forward / Backward / Turning
        (*Right / Left / In-Place / Curved*)
- Multitasking
  - Tasks, Processes, and Threads
  - Parallel and Concurrent processing
  - MicroPython:
    - Threading (…more like “Multicore Programming”)
    - Asynchronous I/O scheduler (AsyncIO)
    - Timers
    - Interrupt Service Routine (ISR)
- Midterm evaluation of the course
- Introduction to Portfolio 2: Differential Drive class
  - No exercises…
  - Extra Point Activity #2: 1.1 and **Check that your videos are public**!

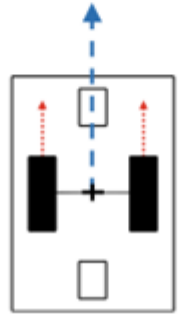
Turning clockwise

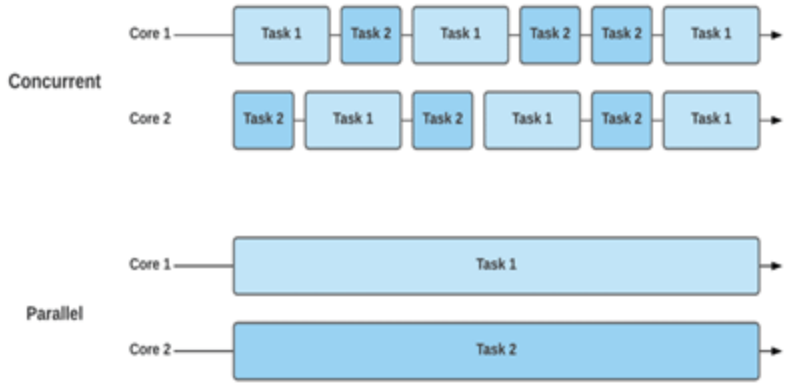


Turning Right (curved)



Driving forward





Sensors

Sensors

*"**Devices** or components that d**etect changes in physical conditions** or environmental variables and convert them into signals that can be measured, recorded, or analyzed."*

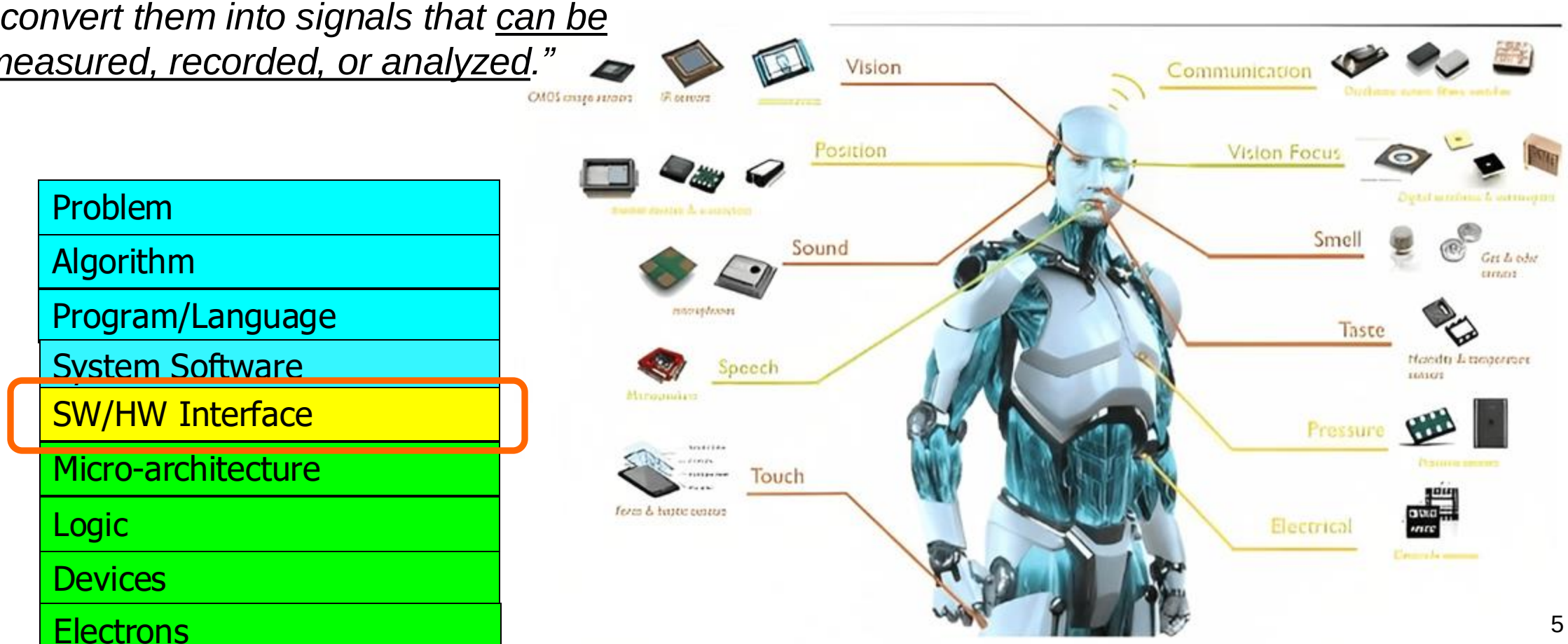


Illustration: https://www.mdpi.com/2218-6581/12/3/68

Sensors

"Devices or components that detect changes in physical conditions or environmental variables and convert them into signals that can be measured, recorded, or analyzed."

…in a mobile robot context

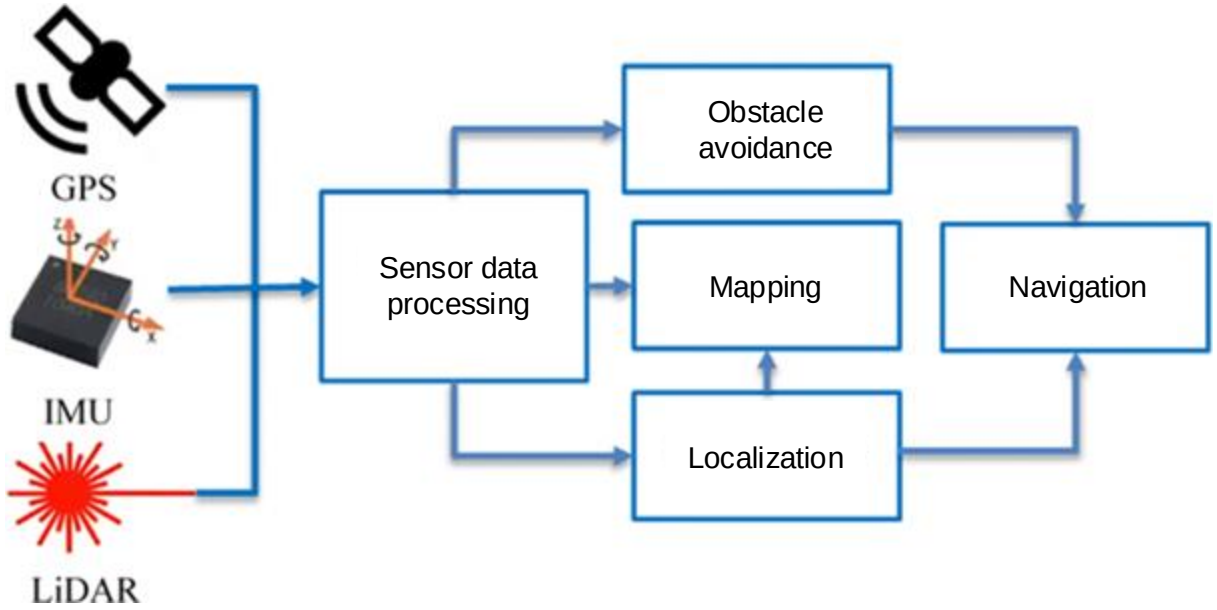


Illustration: Modified from "A robust autonomous navigation and mapping system based on GPS and LiDAR data for unconstraint environment"

Sensors

"Devices or components that detect changes in physical conditions or environmental variables and convert them into signals that can be measured, recorded, or analyzed."

…in a mobile robot context

- **Localization:** "where am I?"

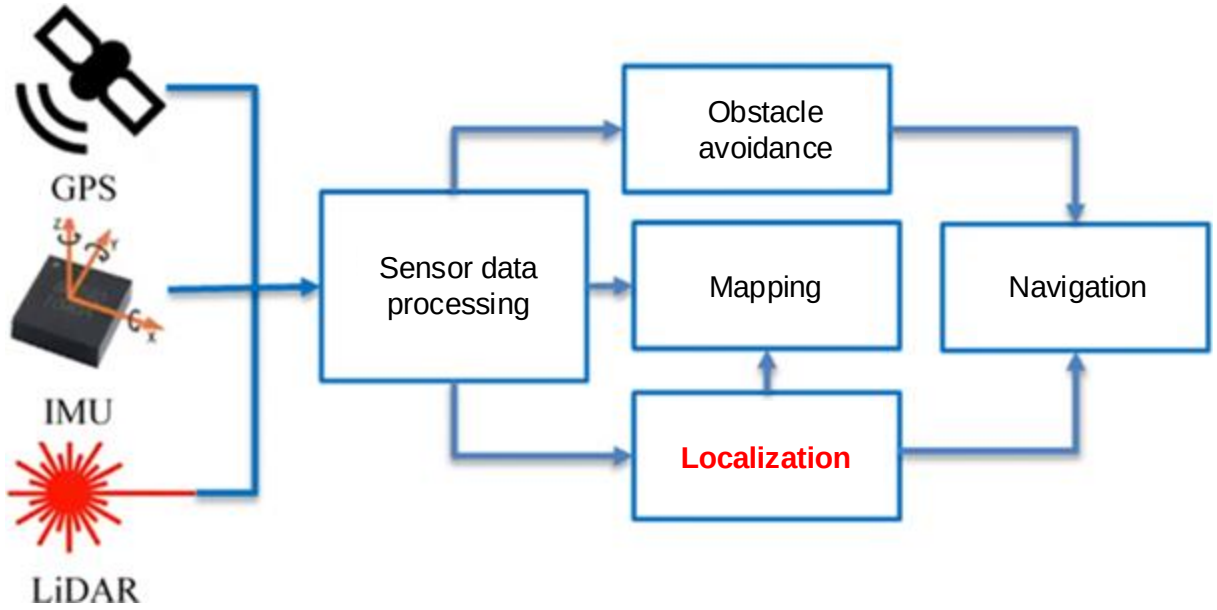


Illustration: Modified from "A robust autonomous navigation and mapping system based on GPS and LiDAR data for unconstraint environment"

Sensors

"Devices or components that detect changes in physical conditions or environmental variables and convert them into signals that can be measured, recorded, or analyzed."

…in a mobile robot context

- **Localization:** "where am I?"
- **Mapping:** "what does the environment look like?"

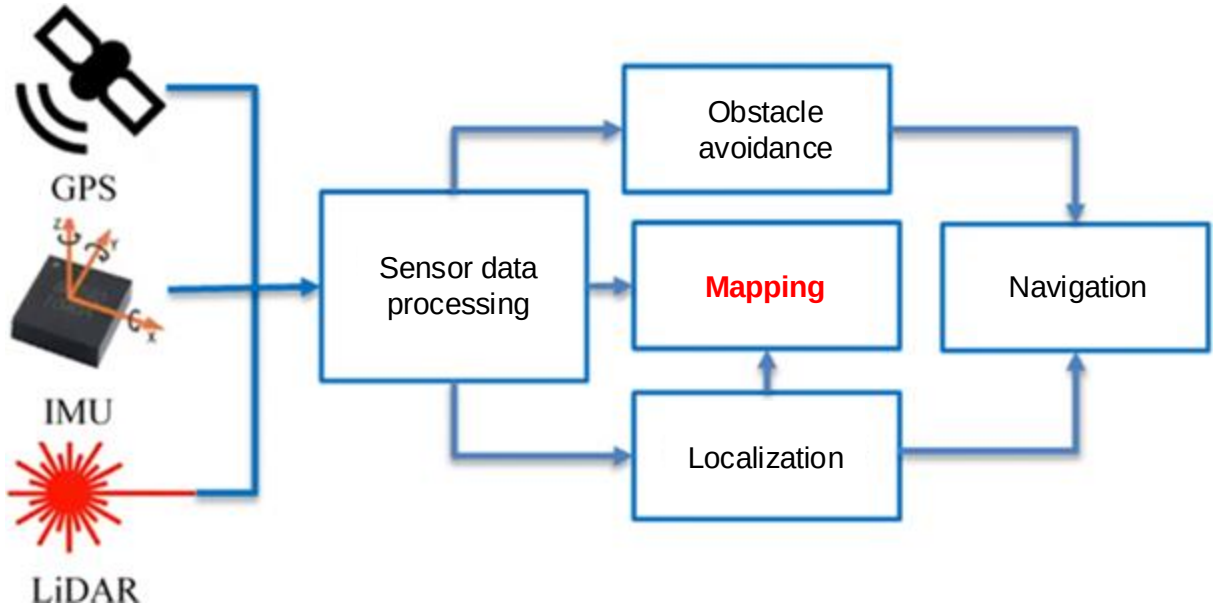


Illustration: Modified from "A robust autonomous navigation and mapping system based on GPS and LiDAR data for unconstraint environment"

Sensors

"Devices or components that detect changes in physical conditions or environmental variables and convert them into signals that can be measured, recorded, or analyzed."

…in a mobile robot context

- **Localization:** "where am I?"
- **Mapping:** "what does the environment look like?"
- **Navigation:** "how do I get there?"

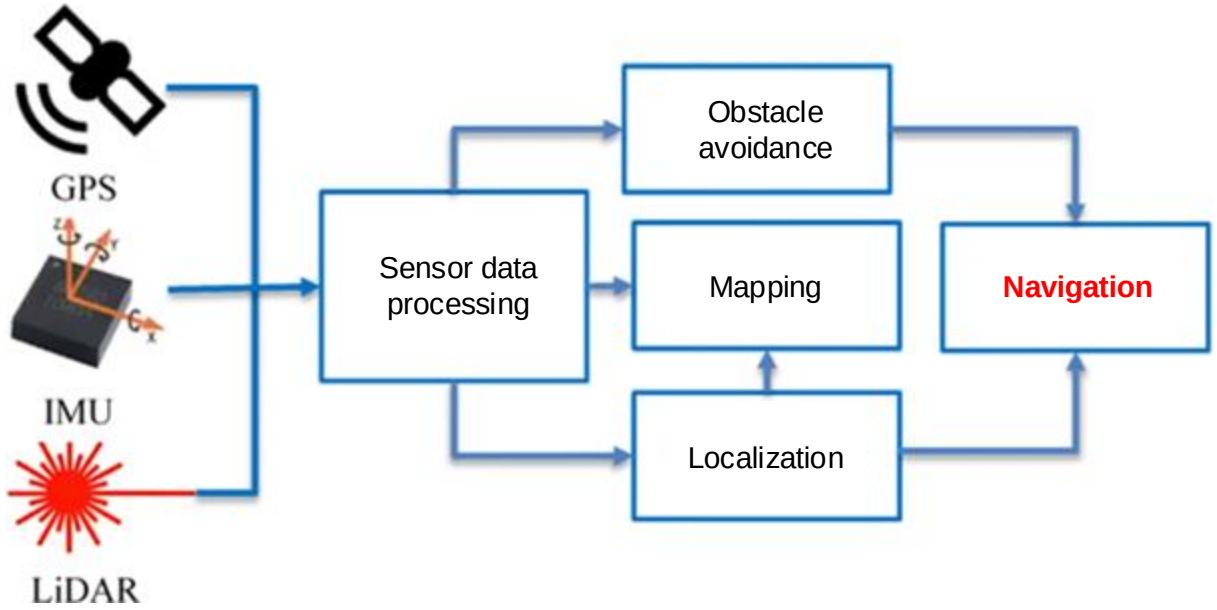


Illustration: Modified from "A robust autonomous navigation and mapping system based on GPS and LiDAR data for unconstraint environment"

Sensors

"Devices or components that detect changes in physical conditions or environmental variables and convert them into signals that can be measured, recorded, or analyzed."

…in a mobile robot context

- **Localization:** "where am I?"
- **Mapping:** "what does the environment look like?"
- **Navigation:** "how do I get there?"
- **Obstacle Avoidance:** "...without hitting any obstacles?"

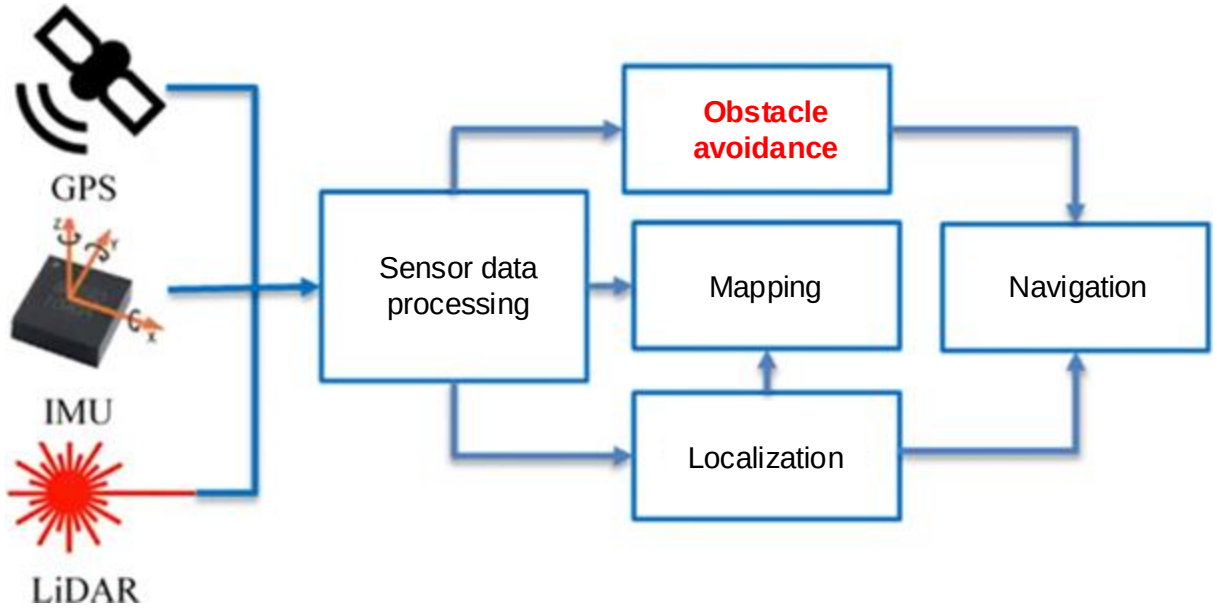


Illustration: Modified from "A robust autonomous navigation and mapping system based on GPS and LiDAR data for unconstraint environment"

Sensors

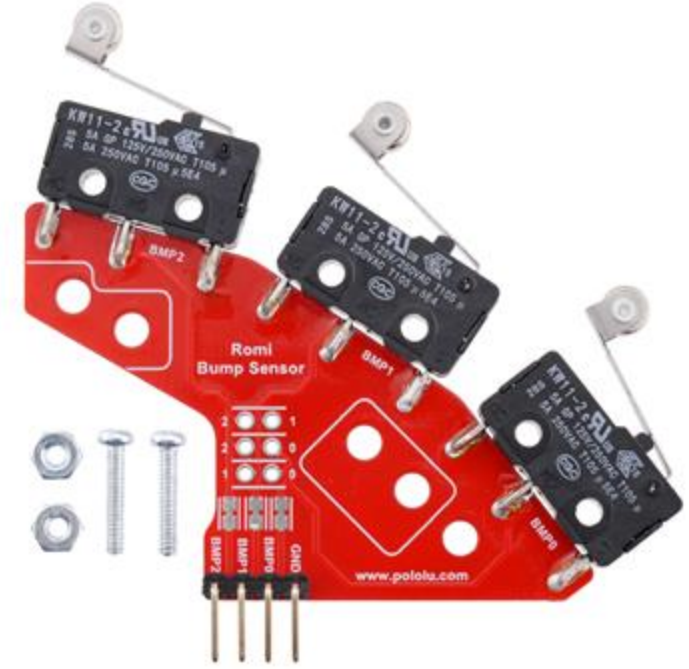
"Devices or components that detect changes in physical conditions or environmental variables and convert them into signals that can be measured, recorded, or analyzed."

Relevant sensor (for mobile robots)

- **Bump sensors** (switches, buttons, etc.)

V-156-1C25

KW12-3



Illustrations: From Google Images

Sensors

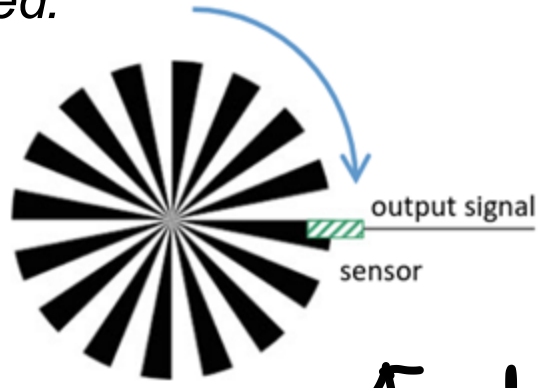
"Devices or components that detect changes in physical conditions or environmental variables and convert them into signals that can be measured, recorded, or analyzed."

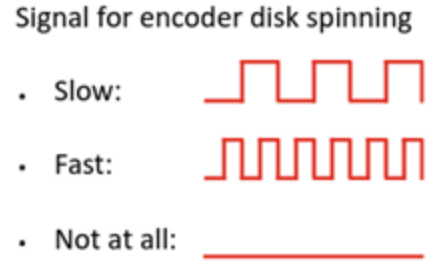
Relevant sensor (for mobile robots)

- **Bump sensors** (switches, buttons, etc.)
- **Rotary/Shaft Encoders** (or counting steps)
  - Measure the rotation of a shaft and are either
    - Incremental (detect relative position / steps),

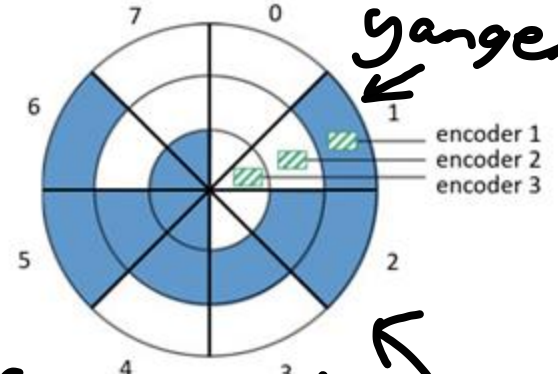
      or

    - Absolute (detect specific position)
      - …using grey code.





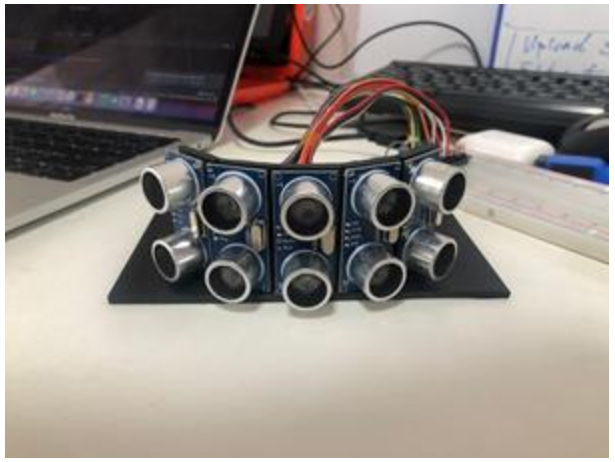
Ændre en bit ad gangen



Grey code

Illustrations: [eB1] From Mobile Robots to Autonomous Vehicles with Raspberry Pi and Arduino

Sensors



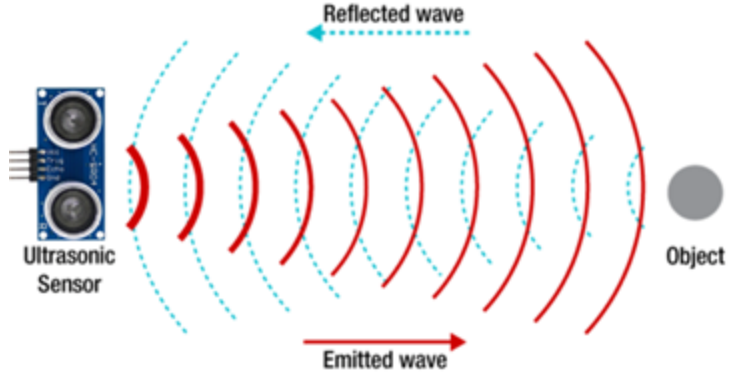
"Devices or components that detect changes in physical conditions or environmental variables and convert them into signals that can be measured, recorded, or analyzed."

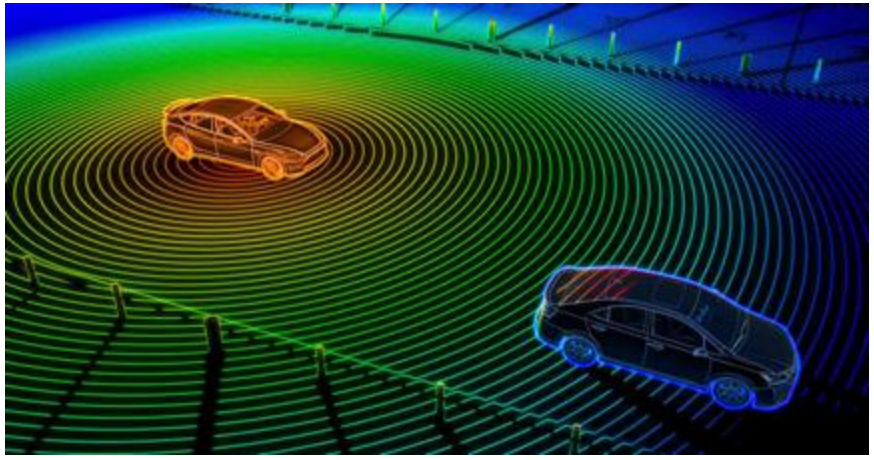
Relevant sensor (for mobile robots)

- **Bump sensors** (switches, buttons, etc.)
- **Rotary/Shaft Encoders** (or counting steps)
- **Distance/range sensors**
  - Sonar, Infrared, and Laser (Time of Flights [ToF])
  - LIDAR (Light Detection and Ranging)

15 hr →

80000 hr ↑

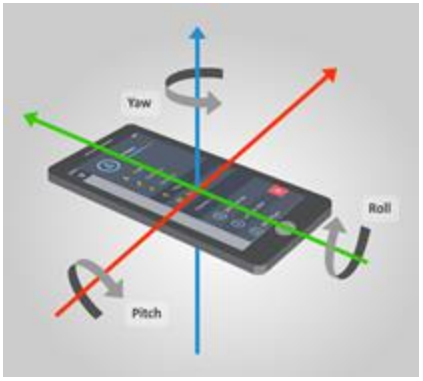




Illustrations: From Google Images

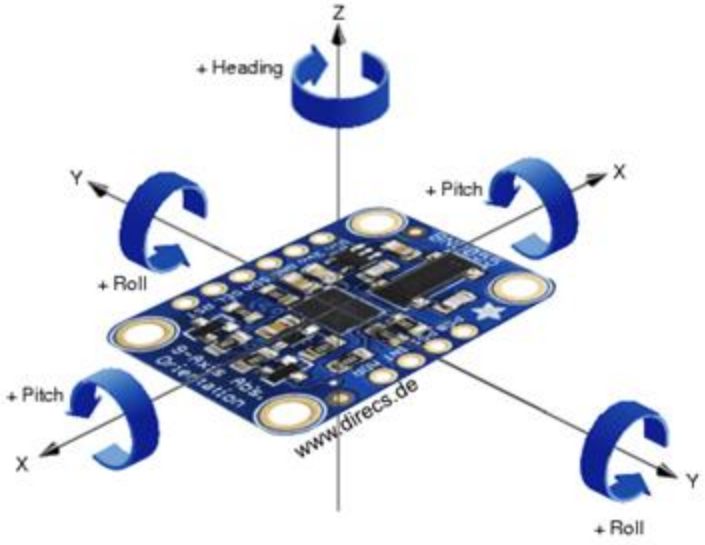
Sensors

"Devices or components that detect changes in physical conditions or environmental variables and convert them into signals that can be measured, recorded, or analyzed."



Relevant sensor (for mobile robots)

- **Bump sensors** (switches, buttons, etc.)
- **Rotary/Shaft Encoders** (or counting steps)
- **Distance/range sensors**
- **Orientation Sensors**
  - **Compass / magnetometer**
    - Measures the Earth's magnetic field (relative orientation)
      - …to help finding the direction relative to magnetic north.
  - **Accelerometers** (1-3 axis)
    - Measures acceleration forces relative to the gravity (movement)
      - …sense changes in speed and direction.
  - **Gyroscopes** (1-3 axis)
    - Measures the angular velocity around an axis (rotation)
      - …sense rotational speed.



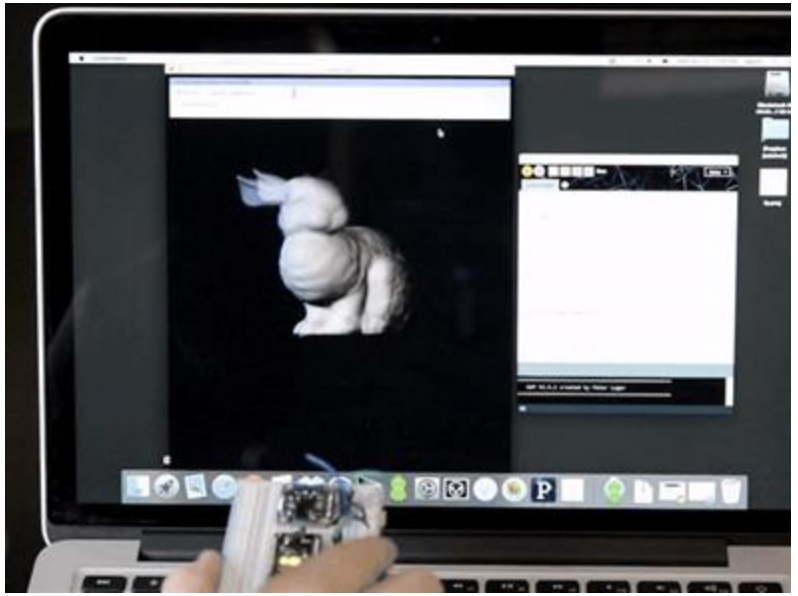
Illustrations: From Google Images

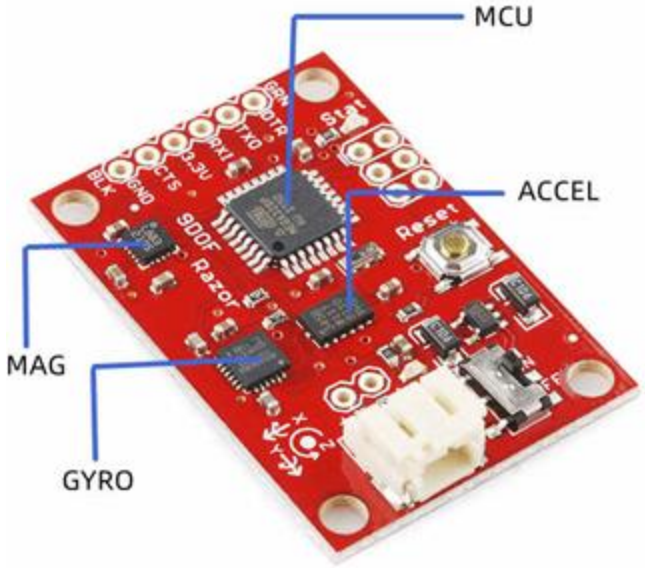
Sensors

"Devices or components that detect changes in physical conditions or environmental variables and convert them into signals that can be measured, recorded, or analyzed."

Relevant sensor (for mobile robots)

- **Bump sensors** (switches, buttons, etc.)
- **Rotary/Shaft Encoders** (or counting steps)
- **Distance/range sensors**
- **Orientation Sensors**
  - Compass / magnetometer
  - Accelerometers
  - Gyroscopes
  - **Inertial Measurement Unit (IMU)**
    - **Combines** accelerometers, gyroscopes, and (sometimes) a magnetometer, to provide a more precise tracking of the motion/position and orientation data.
      - (9) Degree of Freedom [DoF] ?





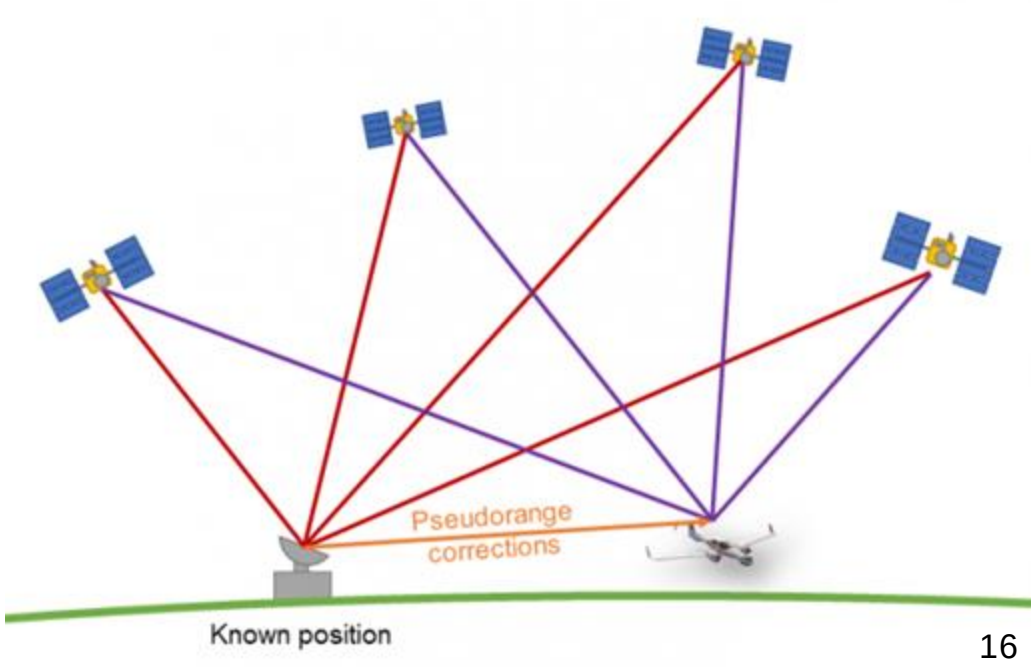
Illustrations: From Google Images

Sensors

"Devices or components that detect changes in physical conditions or environmental variables and convert them into signals that can be measured, recorded, or analyzed."

Relevant sensor (for mobile robots)

- **Bump sensors** (switches, buttons, etc.)
- **Rotary/Shaft Encoders** (or counting steps)
- **Distance/range sensors**
- **Orientation Sensors**
- **Global Navigation Satellite Systems (GNSS)**
  - GPS, GLONASS, Galileo, BeiDou, etc. (2-10 m accuracy)
  - RTK GNSS (1-2 cm accuracy)



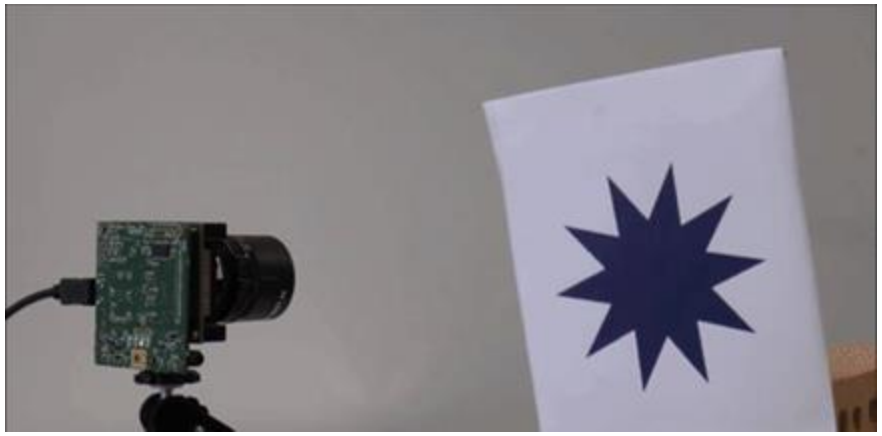
Illustrations: From Google Images

Sensors

"Devices or components that detect changes in physical conditions or environmental variables and convert them into signals that can be measured, recorded, or analyzed."

Relevant sensor (for mobile robots)

- **Bump sensors** (switches, buttons, etc.)
- **Rotary/Shaft Encoders** (or counting steps)
- **Distance/range sensors**
- **Orientation Sensors**
- **Global Navigation Satellite Systems (GNSS)**
- **Vision/Camera Sensors**
  - Mono/Stereo (RGB / Color)
  - Infrared (IR)
  - Thermal
  - Event-based
  - …ect…



Illustrations: From Google Images

Sensors

“Devices or components that detect changes in physical conditions or environmental variables and convert them into signals that can be measured, recorded, or analyzed.”

Sensor categories

- **Internal** versus **External**
- **Local** versus **Global**
- **Active** versus **Passive**

Sensors

"Devices or components that detect changes in physical conditions or environmental variables and convert them into signals that can be measured, recorded, or analyzed."

Sensor categories

- **Internal** versus **External:** The placement of the sensor relative to the system.
    - Internal sensors: monitoring the robot's internal states.
        - **Examples:** battery monitor, motor encoders, IMU
    - External sensors: monitoring the robot's environment. Allows the robot to perceive and interact with its external environment.
        - **Examples:** Sonar sensors, cameras, Lidar

Sensors

"Devices or components that detect changes in physical conditions or environmental variables and convert them into signals that can be measured, recorded, or analyzed."

Sensor categories

- **Local** versus **Global:** The scope or range of the area that the sensor monitors or influences.
    - Local sensors: mounted on the robot.
        - **Examples:** Cameras, IMU, Lidar
    - Global sensors: mounted in its environment and transmitting sensor data back to the robot.
        - **Examples:** Global Navigation Satellite System (GNSS), magnetometer/compass

Sensors

"Devices or components that detect changes in physical conditions or environmental variables and convert them into signals that can be measured, recorded, or analyzed."

Sensor categories

- **Active** versus **Passive:** TheIn te scope or range of the area that the sensor monitors or influences.

    - Active sensors: emits energy (such as light, sound, or electromagnetic waves) and then measure how it is reflected or otherwise affected by the environment.
        - **Examples:** Ultrasonic, IR, LIDAR

    - Passive sensors: rely on detecting energy that is already present in the environment.
        - **Examples:** Cameras, microphone, thermocouples, Light Dependent Resistor (LDr)

Sensors

“Devices or components that detect changes in physical conditions or environmental variables and convert them into signals that can be measured, recorded, or analyzed.”

Sensor types

- **Binary sensors** (signals, either TRUE (1) or FALSE (0))
- **Analog sensors**(signals, e.g. 0 … 5V)
- **Digital sensors** (signals/protocols, e.g. UART, I2C, SPI)

Binary sensors

Binary Sensors

*"A type of sensor that only return a single bit of information, either a **TRUE (1)** or **FALSE (0)**"*

- The simplest type of sensors
  - Interface: GPIOs (Digital I/Os)

- **Example:** Limit Switch for Collision Detection
  - The switch outputs a digital signal:
    - **LOW (0)** when the switch is open (no collision), and
    - **HIGH (1)** when the switch is pressed (collision detected).

  - **Purpose:** The robot detects collisions and can stop or reverse direction upon impact with an object.

KW12-3

V-156-1C25

Illustration: https://newbiely.com/tutorials/raspberry-pico/raspberry-pi-pico-limit-switch

Binary Sensors

"A type of sensor that only return a single bit of information, either a TRUE (1) or FALSE (0)."

- The simplest type of sensors
  - Interface: GPIOs (Digital I/Os)

- **Example:** Limit Switch for Collision Detection
  - The switch outputs a digital signal:
    - **LOW (0)** when the switch is open (no collision), and
    - **HIGH (1)** when the switch is pressed (collision detected).

  - **Purpose:** The robot detects collisions and can stop or reverse direction upon impact with an object.

Example: Limit Switch for Collision Detection

```
from machine import Pin # import from machine

# GP9 with internal pull-down resistor
bumper_switch = Pin("GP9", Pin.IN, Pin.PULL_DOWN)

while True:
    if bumper_switch.value() == 1:
        print("Collision detected!")
        # Code to stop or reverse
    else:
        print("No collision")
        # Code to keep moving
```

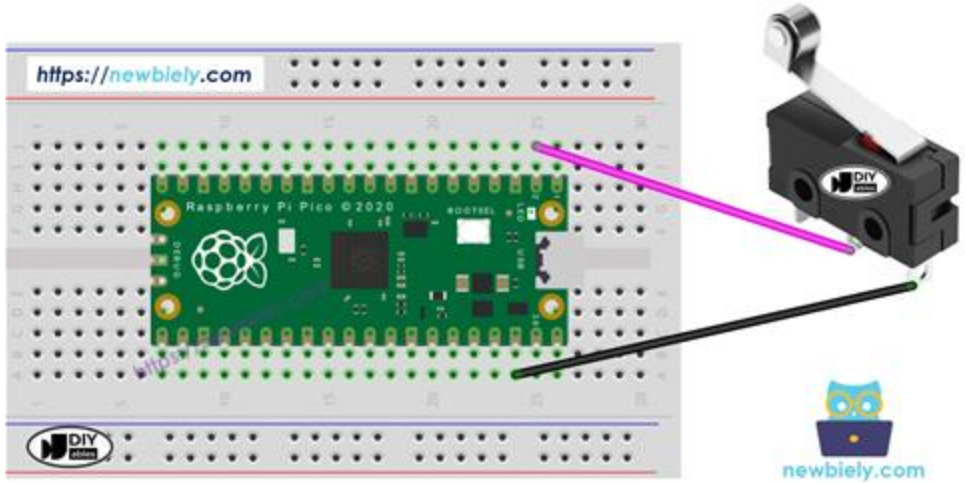


Illustration: https://newbiely.com/tutorials/raspberry-pico/raspberry-pi-pico-limit-switch

Analog sensors
and
Analog-to-Digital Converter (ADC)

Analog sensors

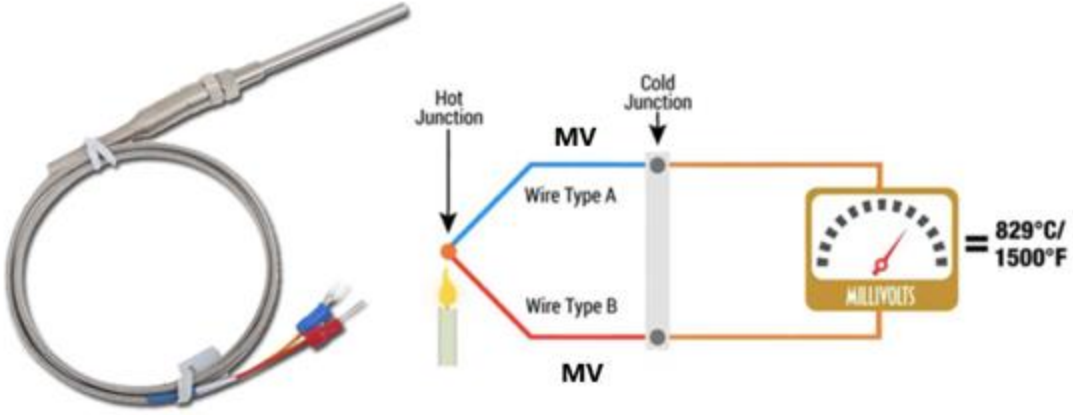
"A type of sensor that generates a continuous, variable electrical signal that corresponds proportionally to a physical quantity it is measuring"

- A number of sensors produce analog output signals rather than digital signals.
  - Usually a *continuous electrical voltage or current*
    - eg. between 0V to 5V
  - Typically converted to a digital value (before processing)

Analog sensors

"A type of sensor that generates a continuous, variable electrical signal that corresponds proportionally to a physical quantity it is measuring"

- **Examples:**
  - **Thermocouples / Temperature sensors**
    - Measure temperature by generating a voltage based on the difference between two metal junctions.

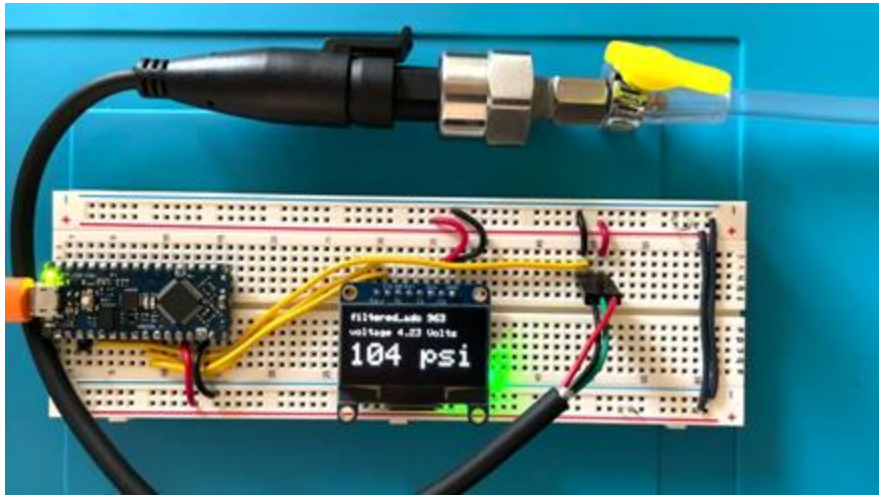


Illustrations: From Google Images

Analog sensors

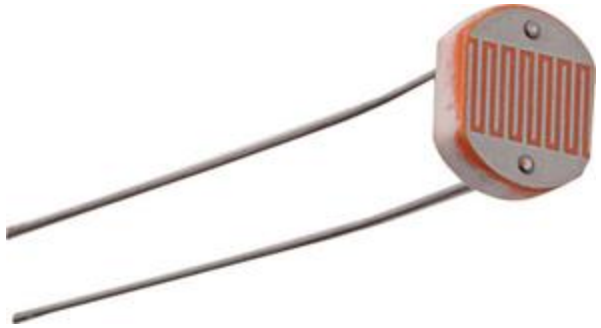
“A type of sensor that generates a continuous, variable electrical signal that corresponds proportionally to a physical quantity it is measuring”

- **Examples:**
  - Thermocouples / Temperature sensors
  - **Pressure**
    - A continuous voltage or current proportional to the pressure level.



Illustrations: From Google Images

Analog sensors



“A type of sensor that generates a continuous, variable electrical signal that corresponds proportionally to a physical quantity it is measuring”

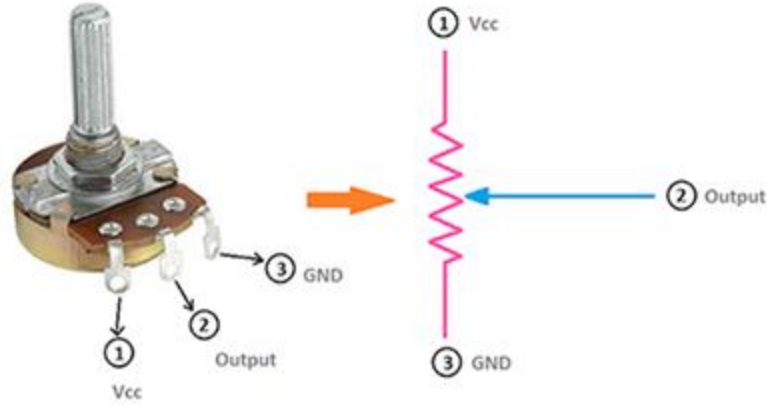
- **Examples:**
  - Thermocouples / Temperature sensors
  - Pressure
  - **Light Dependent Resistor (LDR)**
    - Changes its resistance based on light intensity.
    - Can be used for **automatic light control** or **line detections**

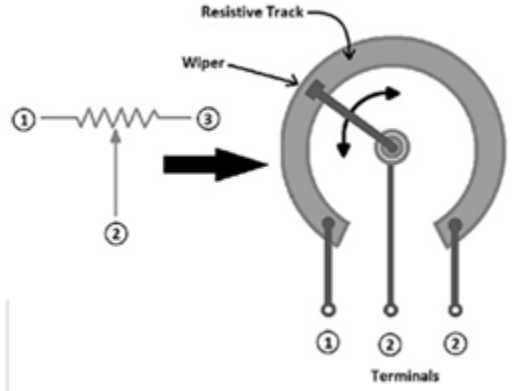
Illustrations: From Google Images

Analog sensors

"A type of sensor that generates a continuous, variable electrical signal that corresponds proportionally to a physical quantity it is measuring"

- **Examples:**
  - Thermocouples / Temperature sensors
  - Pressure
  - Light Dependent Resistor (LDR)
  - **Potentiometers**
    - Variable resistors that change resistance based on position or rotation.
    - Can be used as **position sensors** or **tuning in circuits**

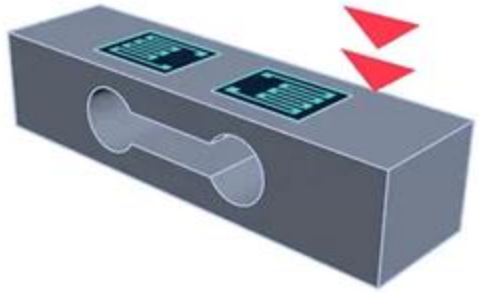




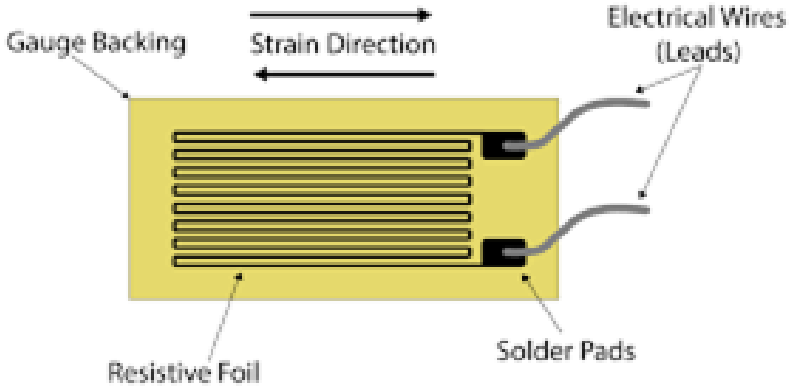
Illustrations: From Google Images

Analog sensors

"A type of sensor that generates a continuous, variable electrical signal that corresponds proportionally to a physical quantity it is measuring"



- **Examples:**
  - Thermocouples / Temperature sensors
  - Pressure
  - Light Dependent Resistor (LDR)
  - Potentiometers
  - **Strain Gauges**
    - Measure deformation (strain) by changing resistance when stretched or compressed.



Illustrations: From Google Images

Analog sensors

"A type of sensor that generates a continuous, variable electrical signal that corresponds proportionally to a physical quantity it is measuring"

- **Examples:**
  - Thermocouples / Temperature sensors
  - Pressure
  - Light Dependent Resistor (LDR)
  - Potentiometers
  - Strain Gauges
  - **…and many more…**

Illustrations: From Google Images

Analog sensors

"A type of sensor that generates a continuous, variable electrical signal that corresponds proportionally to a physical quantity it is measuring"

- **Examples:**
  - Thermocouples
  - Pressure
  - Light Dependent Resistor (LDR)
  - Potentiometers
  - Strain Gauges
  - **…and many more…**

How to convert this into something the microcontroller can process?

Analog sensors

"A type of sensor that generates a continuous, variable electrical signal that corresponds proportionally to a physical quantity it is measuring"

- **Examples:**
  - Thermocouples
  - Pressure
  - Light Dependent Resistor (LDR)
  - Potentiometers
  - Strain Gauges
  - **…and many more…**

How to convert this into something the microcontroller can process?

…use an Analog-to-Digital Converter (ADC)

Analog-to-Digital Converter (ADC)

*"A **device** or peripheral that converts an analog signal (continuous) into a digital value (discrete)."*

Illustration: https://www.seeedstudio.com/blog/2020/10/20/analog-to-digital-converter-adc-reading-raspberry-pi-analog/

Analog-to-Digital Converter (ADC)

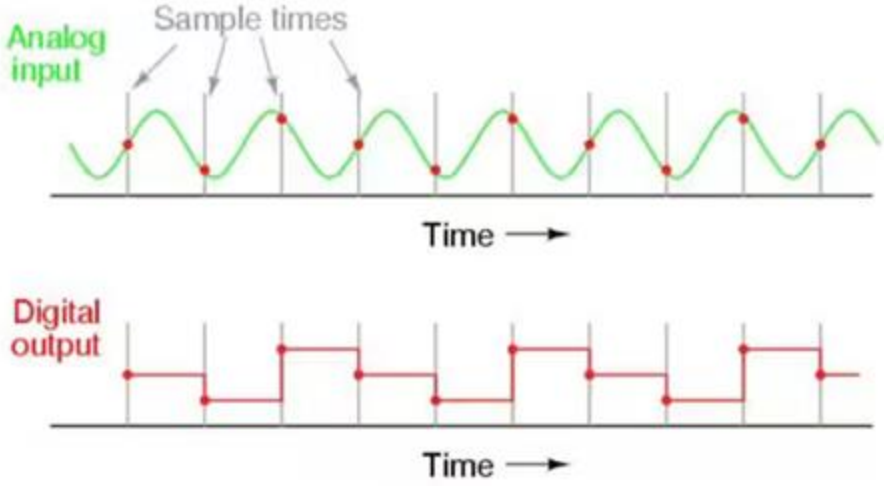
"A device or peripheral that converts an analog signal (continuous) into a digital value (discrete)."

- **Sampling rate:** Measured by samples per second.
  - Typically Samples Per Second (SPS) or Hertz (Hz).

- **Resolution:** How many distinct values (quantization levels) the ADC can output for each sample.
  - Typically specified in bits.

  - **Common Resolutions** (precision):
    - 8-bit ADC: 256 levels
    - 10-bit ADC: 1024 levels
    - 12-bit ADC: 4096 levels
    - 16-bit ADC: 65,536 levels

Sampling rate



Resolution

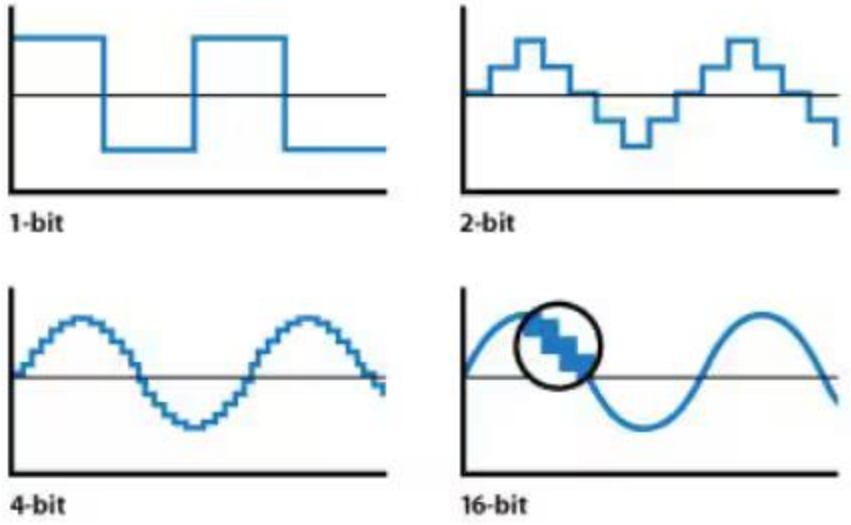


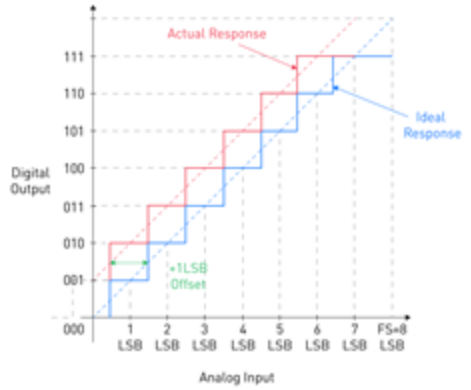
Illustration: https://www.seeedstudio.com/blog/2020/10/20/analog-to-digital-converter-adc-reading-raspberry-pi-analog/

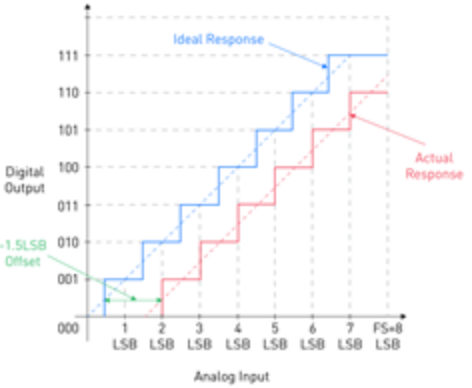
Analog-to-Digital Converter (ADC)

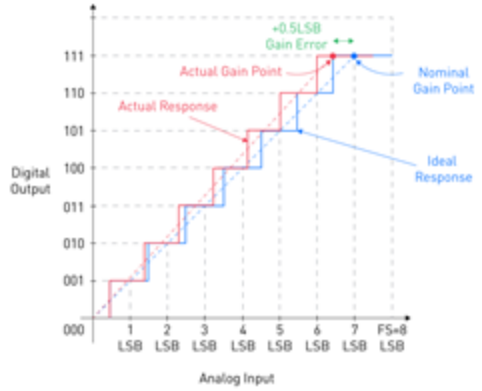
"A device or peripheral that converts an analog signal (continuous) into a digital value (discrete)."

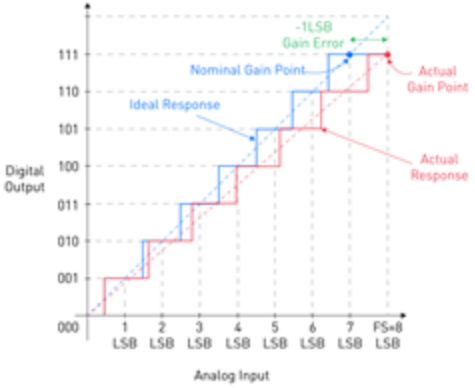
Error sources

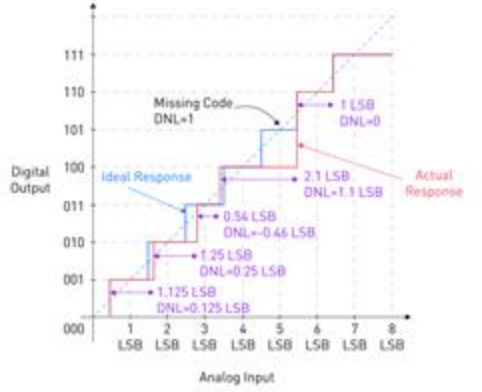
- …an ADC introduce errors that affect the accuracy and precision of the conversion..
  - Most common ones:
    - Offset Errors
    - Gain errors
    - Linearity Errors
    - Quantization errors
    - Aliasing (Sampling errors)
    - Noise
    - Jitter
    - …etc.











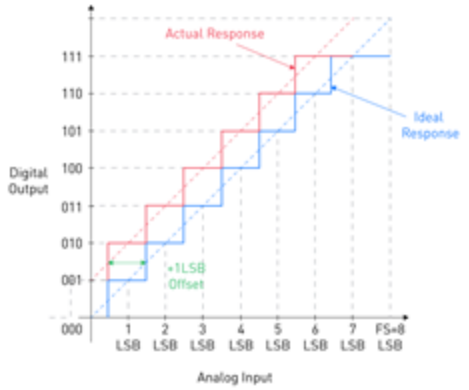
Source: https://www.monolithicpower.com/en/learning/mpscholar/analog-to-digital-converters/adc-errors-and-compensation/sources-of-errors-in-adcs

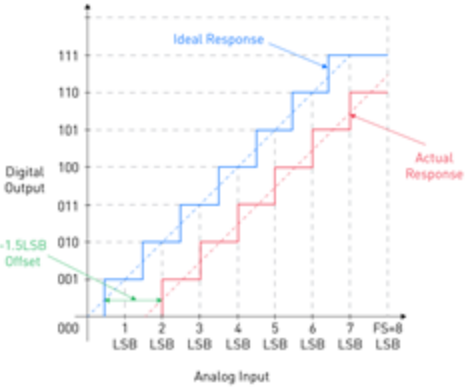
Analog-to-Digital Converter (ADC)

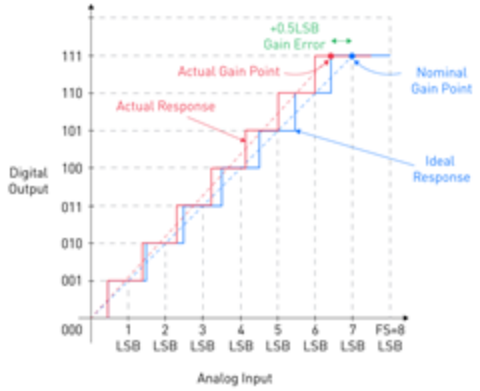
"A device or peripheral that converts an analog signal (continuous) into a digital value (discrete)."

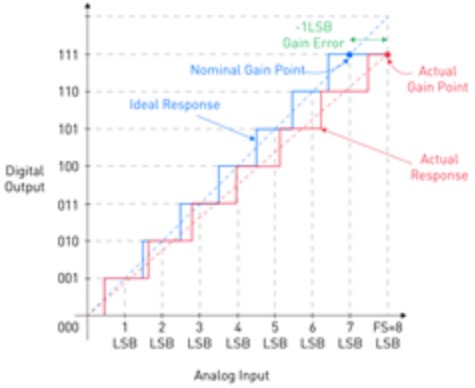
Error sources

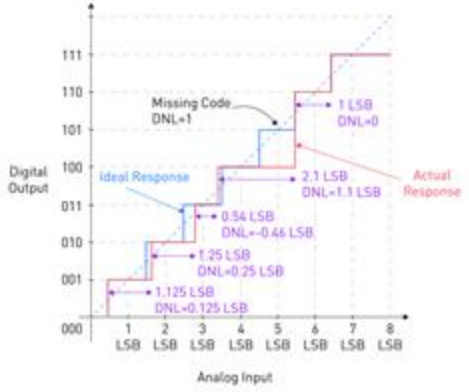
- …an ADC introduce errors that affect the accuracy and precision of the conversion..
  - Most common ones:
    - ~~Offset Errors~~
    - ~~Gain errors~~
    - ~~Linearity Errors~~
    - **Quantization errors**
    - **Aliasing (Sampling errors)**
    - ~~Noise~~
    - ~~Jitter~~
    - ~~…etc.~~







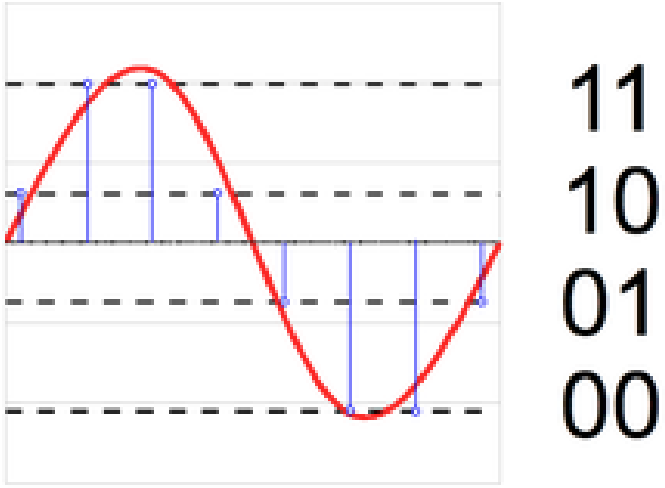




Source: https://www.monolithicpower.com/en/learning/mpscholar/analog-to-digital-converters/adc-errors-and-compensation/sources-of-errors-in-adcs

Analog-to-Digital Converter (ADC)

"A device or peripheral that converts an analog signal (continuous) into a digital value (discrete)."



Error sources

- **Quantization error:** …occurs because an ADC can only represent a finite number of levels.
  - When an analog signal is converted to digital, its continuous amplitude is approximated to the nearest available discrete level.
  - The difference between the actual analog value and the quantized digital value is referred to as the **quantization error**.

The smaller the resolution = the larger the quantization error

(…as each step (or quantum) represents a wider range of values.)

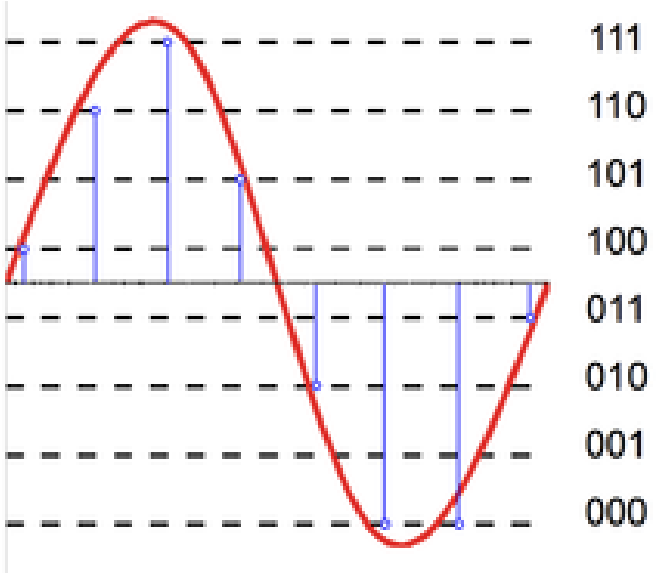
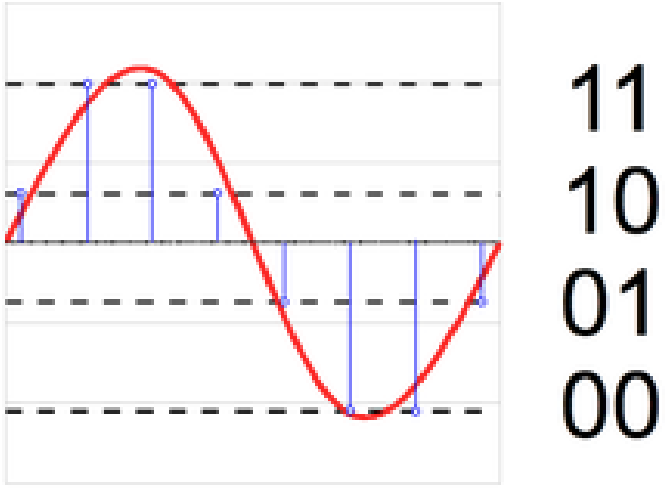


Illustration: https://en.wikipedia.org/wiki/Quantization_%28signal_processing%29

Analog-to-Digital Converter (ADC)

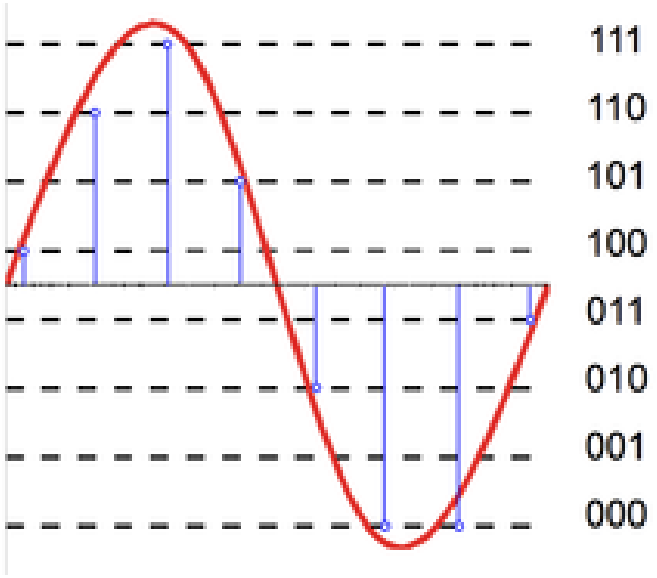
"A device or peripheral that converts an analog signal (continuous) into a digital value (discrete)."



Error sources

- **Quantization error:** …occurs because an ADC can only represent a finite number of levels.
  - When an analog signal is converted to digital, its continuous amplitude is approximated to the nearest available discrete level.
  - The difference between the actual analog value and the quantized digital value is referred to as the **quantization error**.

The smaller the resolution = the larger the quantization error
(…as each step (or quantum) represents a wider range of values.)



- *Example: a 10-bit ADC has 1024 discrete levels, while a 12-bit ADC has 4096 levels, offering finer resolution. (See previous slide)*

Illustration: https://en.wikipedia.org/wiki/Quantization_%28signal_processing%29

Analog-to-Digital Converter (ADC)

"A device or peripheral that converts an analog signal (continuous) into a digital value (discrete)."

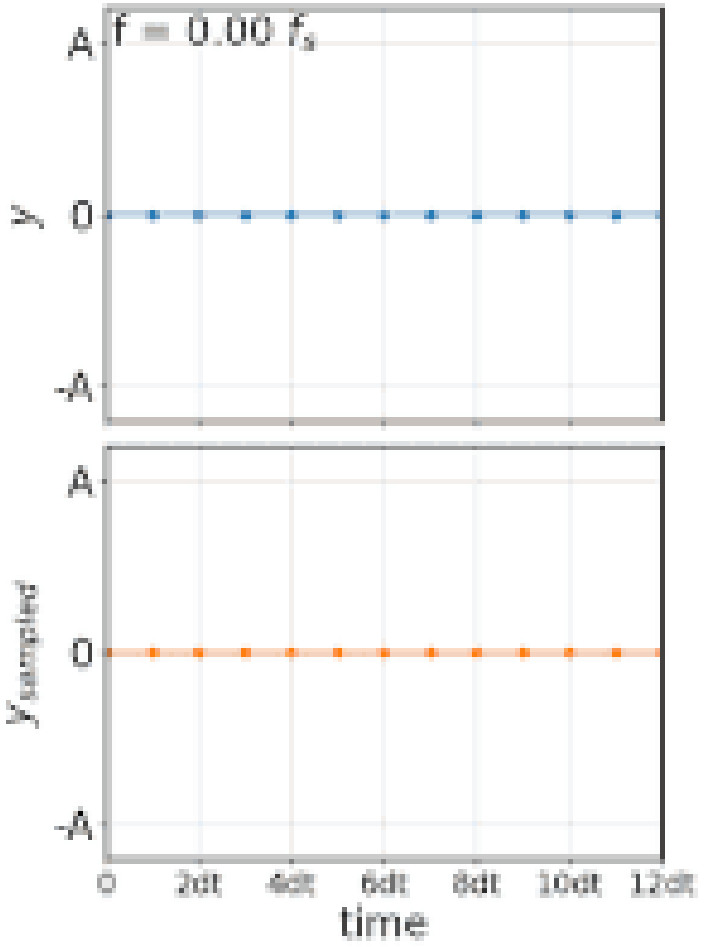
Error sources

- **Aliasing: …**occurs when the ADC samples the analog signal at a rate that is too low, resulting in an inaccurate representation of the signal (overlapping of frequency).
  - **Nyquist rate**: The sampling rate must be at least twice the highest frequency component of the analog signal to accurately reconstruct it:

$$f_s \geq 2 \cdot f_{\max}$$

where
- f_s the sampling rate
  - The rate at which the signal should be sampled
- f_{max} is the highest frequency component present in the analog signal.

"Depicts a sequence of sinusoids, each with a higher frequency than the previous ones. These "true" signals are also being sampled (blue dots) at a constant frequency/rate." [1]



"Using the same samples (now in orange), the default reconstruction algorithm produces the lower-frequency sinusoid." [1]

Illustration: https://en.wikipedia.org/wiki/Aliasing and [1]

Analog-to-Digital Converter (ADC)

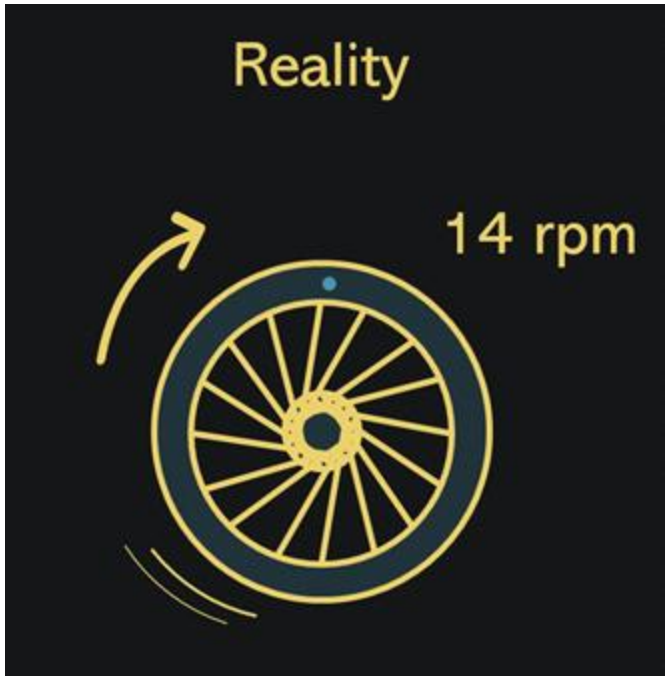
"A device or peripheral that converts an analog signal (continuous) into a digital value (discrete)."

Error sources

- **Aliasing: …**occurs when the ADC samples the analog signal at a rate that is too low, resulting in an inaccurate representation of the signal (overlapping of frequency).
  - **Nyquist rate**: The sampling rate must be at least twice the highest frequency component of the analog signal to accurately reconstruct it:

$$f_s \geq 2 \cdot f_{\max}$$

Example: Recording a spinning wheel



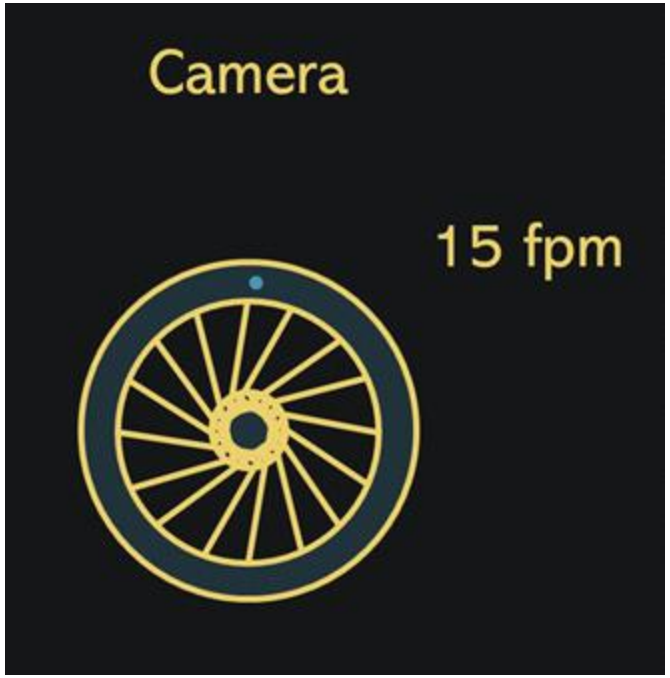
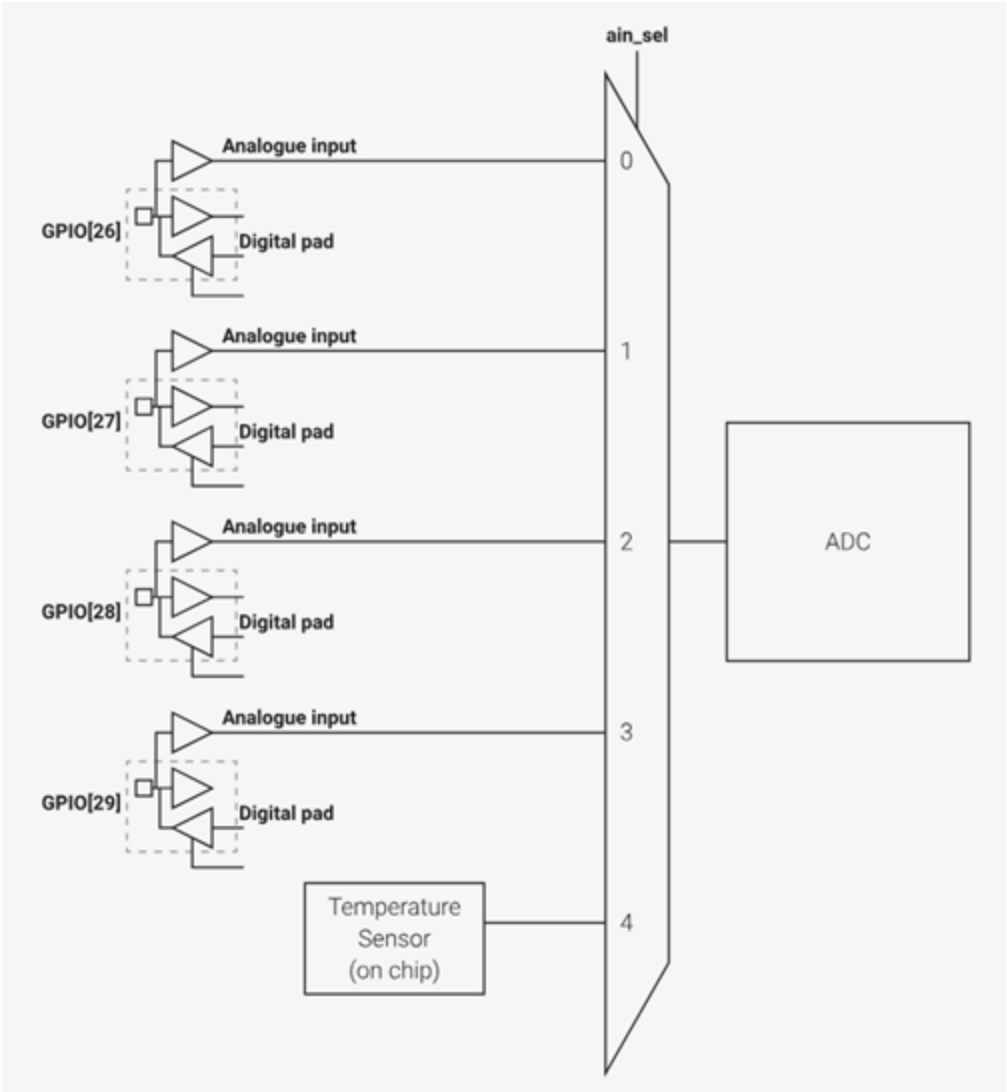


Illustration: https://www.cantorsparadise.com/the-optical-trickery-of-spinning-wheels-9affe410e6a0

Analog-to-Digital Converter (ADC)

"A device or peripheral that converts an analog signal (continuous) into a digital value (discrete)."

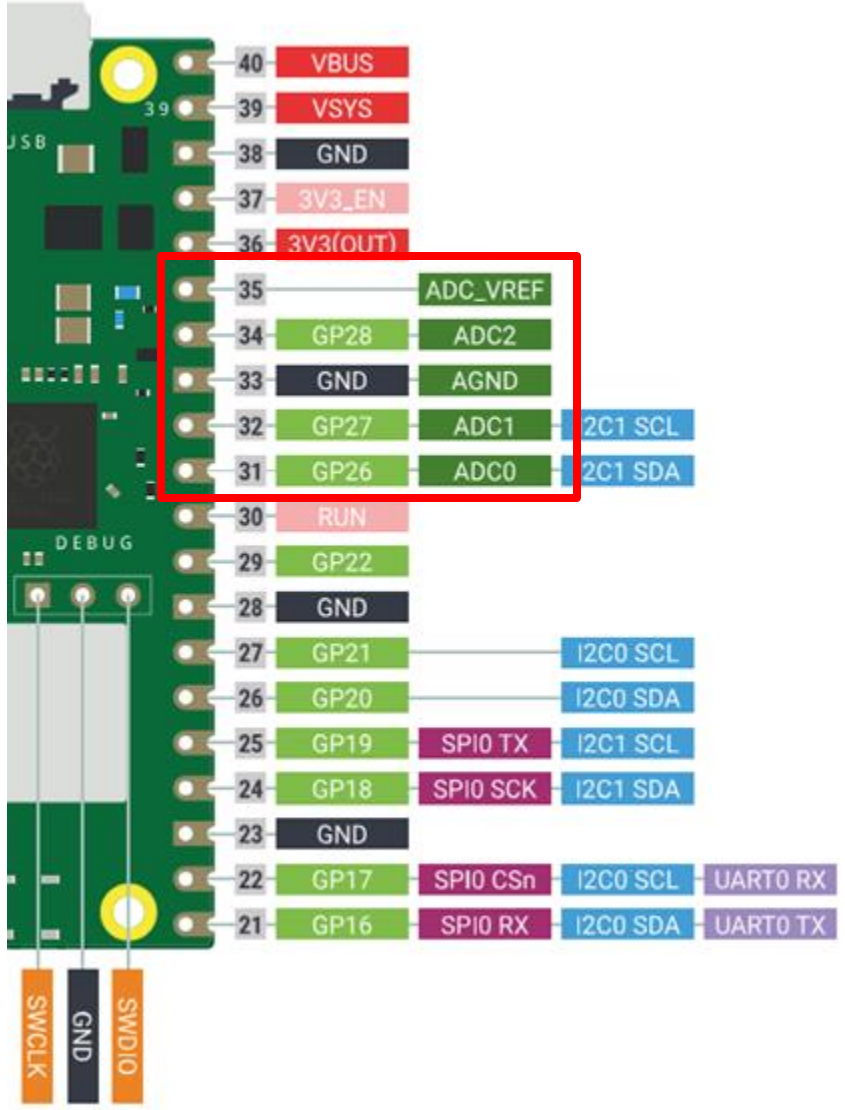
- **ADC Channels on the Pico:**
  - **ADC0**: Pin 31 (GP26)
    - Read analog signals from peripherals
  - **ADC1**: Pin 32 (GP27)
    - Read analog signals from peripherals
  - **ADC2**: Pin 34 (GP28)
    - Read analog signals from peripherals
  - **ADC3:** Pin 35 (GP29)
    - Measure voltage level of VSYS power supply
  - **ADC4:** Internal
    - Read built-in temperature sensor



Analog-to-Digital Converter (ADC)

"A device or peripheral that converts an analog signal (continuous) into a digital value (discrete)."

- **ADC Channels on the Pico:**
  - **ADC0**: Pin 31 (GP26)
    - Read analog signals from peripherals
  - **ADC1**: Pin 32 (GP27)
    - Read analog signals from peripherals
  - **ADC2**: Pin 34 (GP28)
    - Read analog signals from peripherals
  - ~~**ADC3:** Pin 35 (GP29)~~
    - ~~Measure voltage level of VSYS power supply~~
  - ~~**ADC4:** Internal~~
    - ~~Read built-in temperature sensor~~



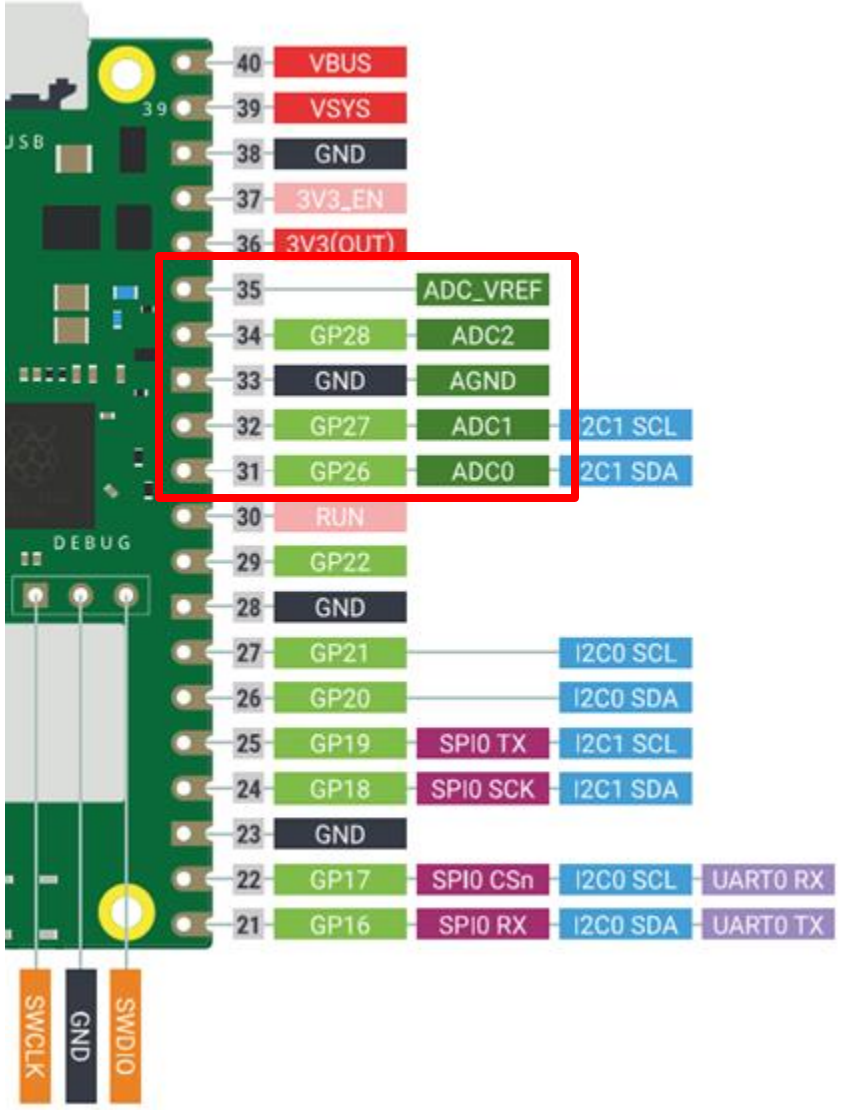
Analog-to-Digital Converter (ADC)

"A device or peripheral that converts an analog signal (continuous) into a digital value (discrete)."

- **Key Features of the Pico's ADC:**
  - **Resolution:** 12-bit resolution
    - corresponds to the range of 0 to 4095 in digital output
  - **Input Voltage Range:** from 0 to 3.3V

$$\text{Voltage Resolution} = \frac{V_{\text{ref}}}{2^n}$$

$$\text{Voltage Resolution} = \frac{3.3V}{2^{12}} = \frac{3.3V}{4096} \approx 0.000805V = 0.805 \text{ mV}$$



Analog-to-Digital Converter (ADC)

"A device or peripheral that converts an analog signal (continuous) into a digital value (discrete)."

- **Example:** Read the internal temperature sensor
  - **According to the RP2040 datasheet,** the formula for converting the ADC voltage to temperature in Celsius is:

$$T(^\circ C) = 27 - \frac{V_{\text{temp}} - 0.706}{0.001721}$$

  - Where
    - V_{temp} is the voltage measured from the temperature sensor
    - 27°C is the temperature at which the voltage is 0.706V
    - 0.001721 is the sensor's temperature coefficient (voltage change per degree Celsius).

Example: Read the value from a LDR

```
from machine import ADC
import time

# Initialize the ADC on the internal temperature sensor (ADC 4)
sensor_temp = ADC(4)

# Reference voltage on the Pico is 3.3V
conversion_factor = 3.3 / (65535)

def read_temperature():
    # Read the raw ADC value (0 - 65535 for a 16-bit reading)
    reading = sensor_temp.read_u16()

    # Convert the raw ADC value to voltage
    voltage = reading * conversion_factor

    # Convert voltage to temperature (degrees Celsius)
    # The formula provided in the RP2040 datasheet:
    # Temperature(°C) = 27 - (Vtemp - 0.706)/0.001721
    temperature = 27 - (voltage - 0.706) / 0.001721

    return temperature

# Example: Continuously read and print the temperatur
while True:
    temperature = read_temperature()
    print("Temperature: {:.2f}°C".format(temperature))
    time.sleep(1)
```

Analog-to-Digital Converter (ADC)

"A device or peripheral that converts an analog signal (continuous) into a digital value (discrete)."

- **Methods**
  - `ADC` **class:** Provides an interface to analog-to-digital converters
    - `class machine.ADC(id, *, sample_ns, atten)`
      - …similar to the `Pin` **class**
  - **`adc.read_u16()`**
    - Reads the ADC value, which ranges from 0 to 65535

Classes (`machine` module)

- `class Pin` – control I/O pins
- `class Signal` – control and sense external I/O devices
- `class ADC` – analog to digital conversion
- `class ADCBlock` – control ADC peripherals
- ...

Example: Read the value from a LDR

```
from machine import ADC
import time

# Initialize the ADC on the internal temperature sensor (ADC 4)
sensor_temp = ADC(4)

# Reference voltage on the Pico is 3.3V
conversion_factor = 3.3 / (65535)

def read_temperature():
    # Read the raw ADC value (0 - 65535 for a 16-bit reading)
    reading = sensor_temp.read_u16()

    # Convert the raw ADC value to voltage
    voltage = reading * conversion_factor

    # Convert voltage to temperature (degrees Celsius)
    # The formula provided in the RP2040 datasheet:
    # Temperature(°C) = 27 - (Vtemp - 0.706)/0.001721
    temperature = 27 - (voltage - 0.706) / 0.001721

    return temperature

# Example: Continuously read and print the temperatur
while True:
    temperature = read_temperature()
    print("Temperature: {:.2f}°C".format(temperature))
    time.sleep(1)
```

Analog-to-Digital Converter (ADC)

"A device or peripheral that converts an analog signal (continuous) into a digital value (discrete)."

- **Methods**
  - `ADC` **class:** Provides an interface to analog-to-digital converters
    - `class machine.ADC(id, *, sample_ns, atten)`
      - …similar to the `Pin` **class**
  - **`adc.read_u16()`**
    - Reads the ADC value, which ranges from 0 to 65535

Classes (`machine` module)

- `class Pin` – control I/O pins
- `class Signal` – control and sense external I/O devices
- `class ADC` – analog to digital conversion
- `class ADCBlock` – control ADC peripherals
- ...

Example: Read the value from a LDR

```
from machine import ADC
import time

# Initialize the ADC on the internal temperature sensor (ADC 4)
sensor_temp = ADC(4)

# Reference voltage on the Pico is 3.3V
conversion_factor = 3.3 / (65535)

def read_temperature():
    # Read the raw ADC value (0 - 65535 for a 16-bit reading)
    reading = sensor_temp.read_u16()

    # Convert the raw ADC value to voltage
    voltage = reading * conversion_factor

    # Convert voltage to temperature (degrees Celsius)
    # The formula provided in the RP2040 datasheet:
    # Temperature(°C) = 27 - (Vtemp - 0.706)/0.001721
    temperature = 27 - (voltage - 0.706) / 0.001721

    return temperature

# Example: Continuously read and print the temperatur
while True:
    temperature = read_temperature()
    print("Temperature: {:.2f}°C".format(temperature))
    time.sleep(1)
```

Analog-to-Digital Converter (ADC)

"A device or peripheral that converts an analog signal (continuous) into a digital value (discrete)."

- **Methods**
  - `ADC` **class:** Provides an interface to analog-to-digital converters
    - `class machine.ADC(id, *, sample_ns, atten)`
      - …similar to the `Pin` **class**
  - **`adc.read_u16()`**
    - Reads the ADC value, which ranges from 0 to 65535

Classes (`machine` module)

- `class Pin` – control I/O pins
- `class Signal` – control and sense external I/O devices
- `class ADC` – analog to digital conversion
- `class ADCBlock` – control ADC peripherals
- ...

Example: Read the value from a LDR

```
from machine import ADC
import time

# Initialize the ADC on the internal temperature sensor (ADC 4)
sensor_temp = ADC(4)

# Reference voltage on the Pico is 3.3V
conversion_factor = 3.3 / (65535)

def read_temperature():
    # Read the raw ADC value (0 - 65535 for a 16-bit reading)
    reading = sensor_temp.read_u16()

    # Convert the raw ADC value to voltage
    voltage = reading * conversion_factor

    # Convert voltage to temperature (degrees Celsius)
    # The formula provided in the RP2040 datasheet:
    # Temperature(°C) = 27 - (Vtemp - 0.706)/0.001721
    temperature = 27 - (voltage - 0.706) / 0.001721

    return temperature

# Example: Continuously read and print the temperatur
while True:
    temperature = read_temperature()
    print("Temperature: {:.2f}°C".format(temperature))
    time.sleep(1)
```

Analog-to-Digital Converter (ADC)

"A device or peripheral that converts an analog signal (continuous) into a digital value (discrete)."

- **Methods**
  - `ADC` **class:** Provides an interface to analog-to-digital converters
    - `class machine.ADC(id, *, sample_ns, atten)`
      - …similar to the `Pin` **class**
  - **`adc.read_u16()`**
    - Reads the ADC value, which ranges from 0 to 65535

Classes (`machine` module)

- `class Pin` – control I/O pins
- `class Signal` – control and sense external I/O devices
- `class ADC` – analog to digital conversion
- `class ADCBlock` – control ADC peripherals
- ...

Example: Read the value from a LDR

```
from machine import ADC
import time

# Initialize the ADC on the internal temperature sensor (ADC 4)
sensor_temp = ADC(4)

# Reference voltage on the Pico is 3.3V
conversion_factor = 3.3 / (65535)

def read_temperature():
    # Read the raw ADC value (0 - 65535 for a 16-bit reading)
    reading = sensor_temp.read_u16()

    # Convert the raw ADC value to voltage
    voltage = reading * conversion_factor

    # Convert voltage to temperature (degrees Celsius)
    # The formula provided in the RP2040 datasheet:
    # Temperature(°C) = 27 - (Vtemp - 0.706)/0.001721
    temperature = 27 - (voltage - 0.706) / 0.001721

    return temperature

# Example: Continuously read and print the temperatur
while True:
    temperature = read_temperature()
    print("Temperature: {:.2f}°C".format(temperature))
    time.sleep(1)
```

Analog-to-Digital Converter (ADC)

"A device or peripheral that converts an analog signal (continuous) into a digital value (discrete)."

- **But…**

12-bit ADC Resolution
==
16-bit Output from `read_u16()`

??

Example: Read the value from a LDR

```
from machine import ADC
import time

# Initialize the ADC on the internal temperature sensor (ADC 4)
sensor_temp = ADC(4)

# Reference voltage on the Pico is 3.3V
conversion_factor = 3.3 / (65535)

def read_temperature():
    # Read the raw ADC value (0 - 65535 for a 16-bit reading)
    reading = sensor_temp.read_u16()

    # Convert the raw ADC value to voltage
    voltage = reading * conversion_factor

    # Convert voltage to temperature (degrees Celsius)
    # The formula provided in the RP2040 datasheet:
    # Temperature(°C) = 27 - (Vtemp - 0.706)/0.001721
    temperature = 27 - (voltage - 0.706) / 0.001721

    return temperature

# Example: Continuously read and print the temperatur
while True:
    temperature = read_temperature()
    print("Temperature: {:.2f}°C".format(temperature))
    time.sleep(1)
```

Analog-to-Digital Converter (ADC)

"A device or peripheral that converts an analog signal (continuous) into a digital value (discrete)."

- **But…**

12-bit ADC Resolution
==
16-bit Output from `read_u16()`

…the 12-bit value is scaled up to fit into a 16-bit number for uniformity across platforms…

Example: Read the value from a LDR

```
from machine import ADC
import time

# Initialize the ADC on the internal temperature sensor (ADC 4)
sensor_temp = ADC(4)

# Reference voltage on the Pico is 3.3V
conversion_factor = 3.3 / (65535)

def read_temperature():
    # Read the raw ADC value (0 - 65535 for a 16-bit reading)
    reading = sensor_temp.read_u16()

    # Convert the raw ADC value to voltage
    voltage = reading * conversion_factor

    # Convert voltage to temperature (degrees Celsius)
    # The formula provided in the RP2040 datasheet:
    # Temperature(°C) = 27 - (Vtemp - 0.706)/0.001721
    temperature = 27 - (voltage - 0.706) / 0.001721

    return temperature

# Example: Continuously read and print the temperatur
while True:
    temperature = read_temperature()
    print("Temperature: {:.2f}°C".format(temperature))
    time.sleep(1)
```

Analog-to-Digital Converter (ADC)

"A device or peripheral that converts an analog signal (continuous) into a digital value (discrete)."

- **Example:** Read the value from a LDR
  - **Purpose:** The device detects if it is too dark or light using the LDR, and controls the blinds accordingly.

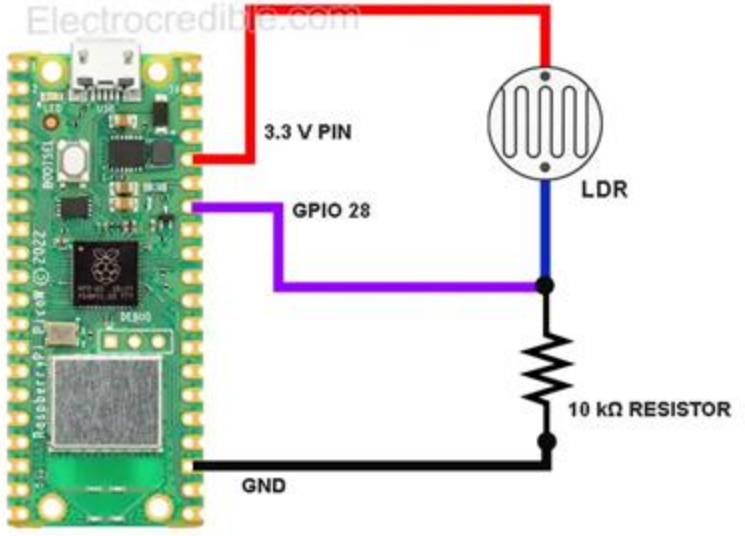


Illustration: https://electrocredible.com/raspberry-pi-pico-ldr-photoresistor-light-dark-sensor/

Example: Read the value from a LDR

```
from machine import ADC  # Import the ADC class from the machine module
import time

adc = ADC(28)  # Initialize the ADC object on pin 28

# The threshold voltage as 50% of the maximum voltage
threshold_voltage = 3.3 / 2

# Continuously read and display voltage
while True:
    # Read the 16-bit digital value from the ADC
    digital_value = adc.read_u16()

    # Convert the digital value to a voltage (0-3.3V)
    volt = 3.3 * (digital_value / 65535)
    print("Voltage", volt)  # Print the voltage

    # Check if the voltage is lower than the threshold
    if volt < threshold_voltage:
        print("Too light!")
        # Code to roll down the blanks
    else:
        print("Too dark!")
        # Code to roll up the blanks

    # Wait for 0.1 seconds before the next reading
    time.sleep(0.1)
```

Analog-to-Digital Converter (ADC)

"A device or peripheral that converts an analog signal (continuous) into a digital value (discrete)."

- **Example:** Read the value from a LDR
  - **Purpose:** The device detects if it is too dark or light using the LDR, and controls the blinds accordingly.

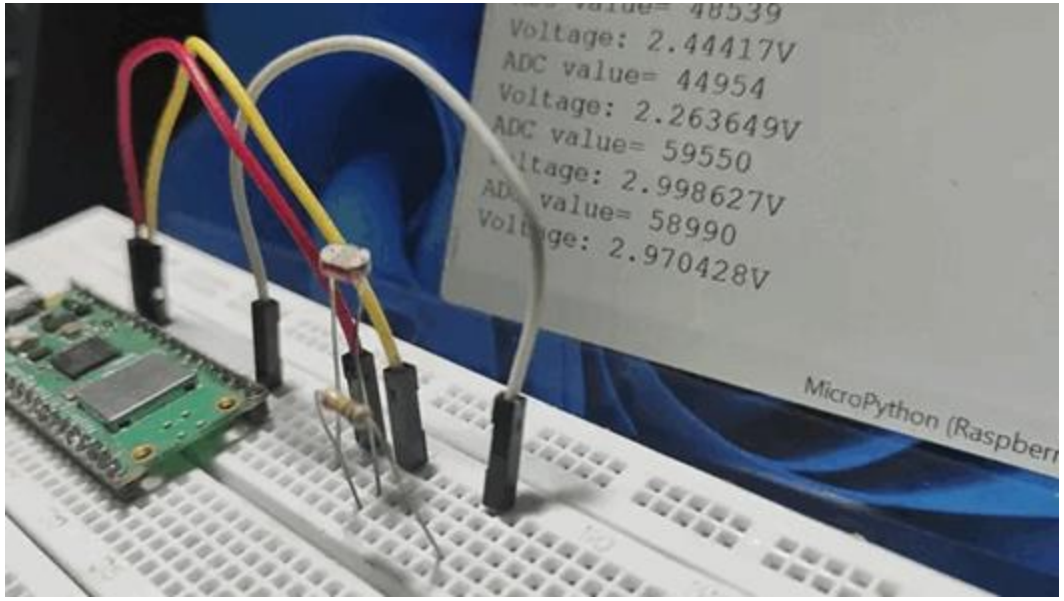


Illustration: https://electrocredible.com/raspberry-pi-pico-ldr-photoresistor-light-dark-sensor/

Example: Read the value from a LDR

```
from machine import ADC  # Import the ADC class from the machine module
import time

adc = ADC(28)  # Initialize the ADC object on pin 28

# The threshold voltage as 50% of the maximum voltage
threshold_voltage = 3.3 / 2

# Continuously read and display voltage
while True:
    # Read the 16-bit digital value from the ADC
    digital_value = adc.read_u16()

    # Convert the digital value to a voltage (0-3.3V)
    volt = 3.3 * (digital_value / 65535)
    print("Voltage", volt)  # Print the voltage

    # Check if the voltage is lower than the threshold
    if volt < threshold_voltage:
        print("Too light!")
        # Code to roll down the blanks
    else:
        print("Too dark!")
        # Code to roll up the blanks

    # Wait for 0.1 seconds before the next reading
    time.sleep(0.1)
```

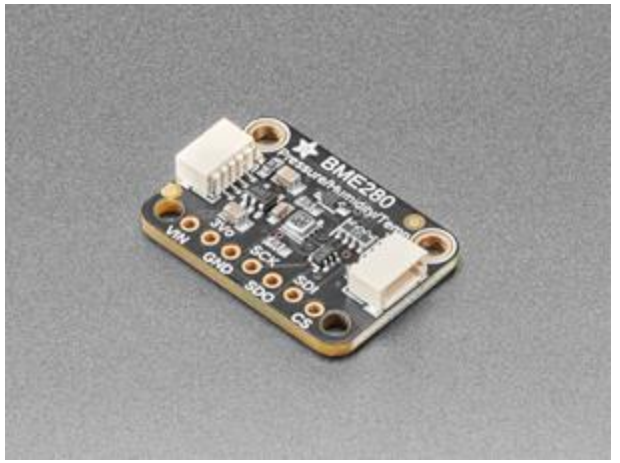
Digital sensors

Digital sensors

"A type of sensor that directly outputs discrete digital data, typically in the form of binary signals (0s and 1s), representing the physical quantity it measures."

- …usually include an integrated ADC
  - Converts the sensor's analog signal into a digital output or data.

Digital sensors



"A type of sensor that directly outputs discrete digital data, typically in the form of binary signals (0s and 1s), representing the physical quantity it measures."

- **…**usually include an integrated ADC
  - Converts the sensor's analog signal into a digital output or data.

- …usually more complex and expensive.
  - **BME280** (Temperature, Humidity, and Pressure Sensor)

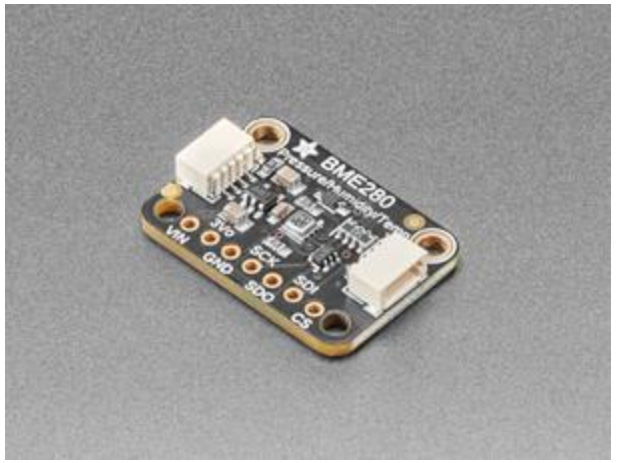
Illustration: https://www.adafruit.com

Digital sensors

"A type of sensor that directly outputs discrete digital data, typically in the form of binary signals (0s and 1s), representing the physical quantity it measures."

- **…**usually include an integrated ADC
  - Converts the sensor's analog signal into a digital output or data.

- …usually more complex and expensive.
  - **BME280** (Temperature, Humidity, and Pressure Sensor)
  - **BNO085** (9-DOF Orientation IMU Fusion Breakout)



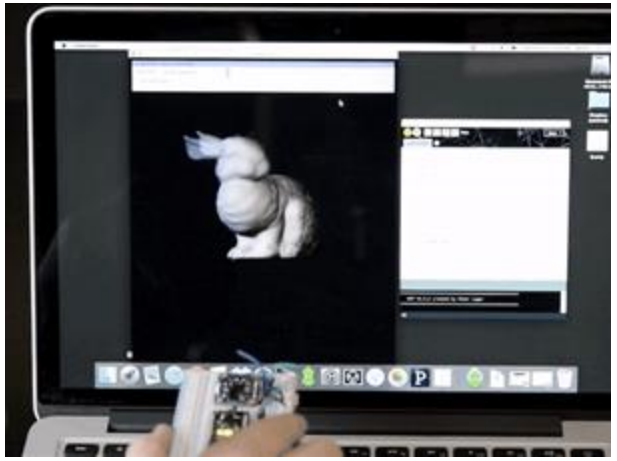


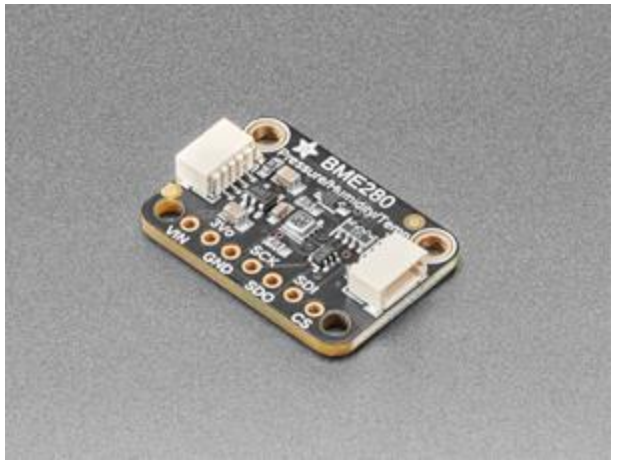
Illustration: https://www.adafruit.com

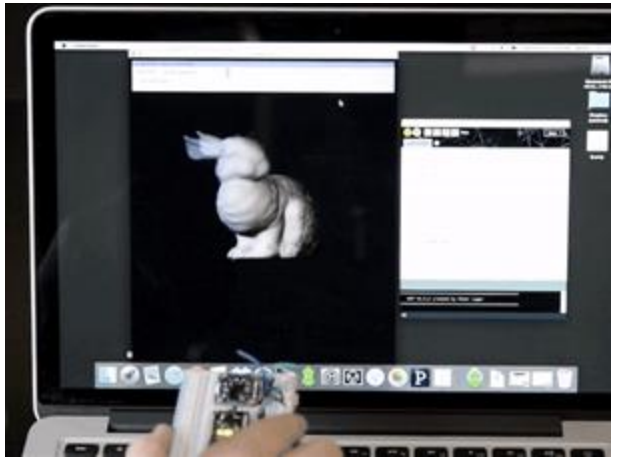
Digital sensors

"A type of sensor that directly outputs discrete digital data, typically in the form of binary signals (0s and 1s), representing the physical quantity it measures."

- **…**usually include an integrated ADC
  - Converts the sensor's analog signal into a digital output or data.

- …usually more complex and expensive.
  - **BME280** (Temperature, Humidity, and Pressure Sensor)
  - **BNO085** (9-DOF Orientation IMU Fusion Breakout)
  - **VL53L1X** (Time of Flight Distance Sensor)





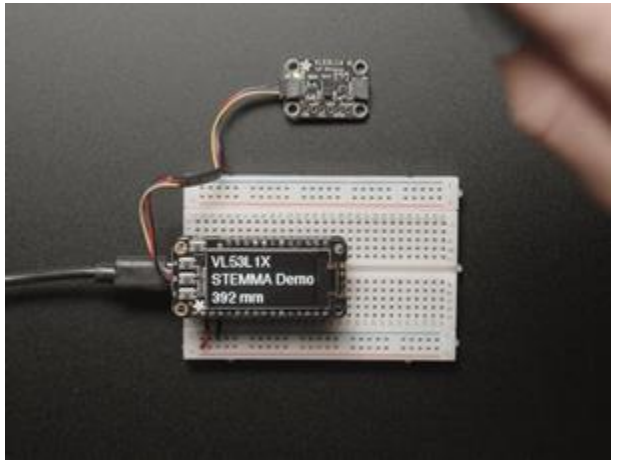


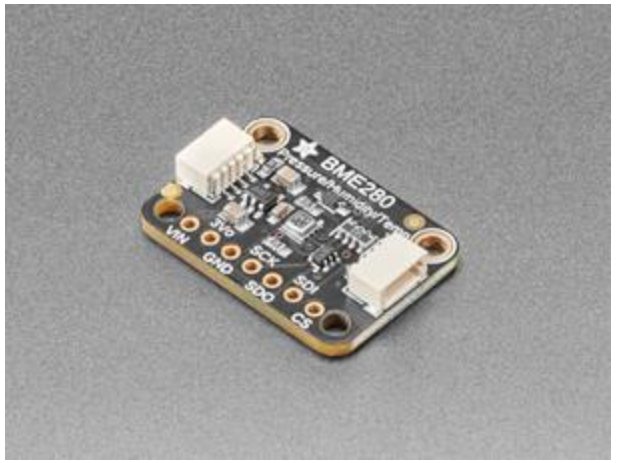
Illustration: https://www.adafruit.com

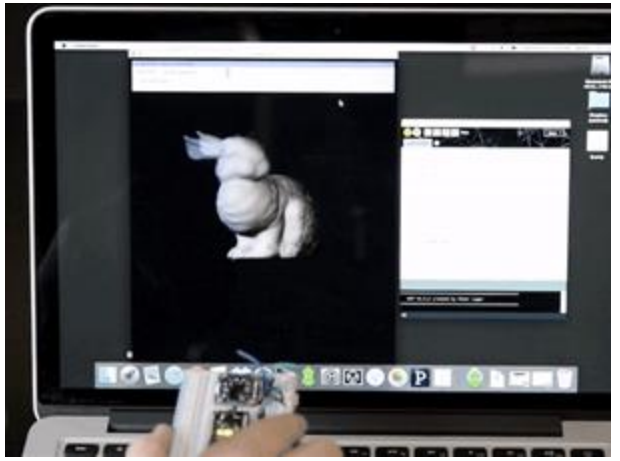
Digital sensors

"A type of sensor that directly outputs discrete digital data, typically in the form of binary signals (0s and 1s), representing the physical quantity it measures."

- …usually include an integrated ADC
  - Converts the sensor's analog signal into a digital output or data.

- …usually more complex and expensive.
  - **BME280** (Temperature, Humidity, and Pressure Sensor)
  - **BNO085** (9-DOF Orientation IMU Fusion Breakout)
  - **VL53L1X** (Time of Flight Distance Sensor)
  - **…and many, many, … , many more…**





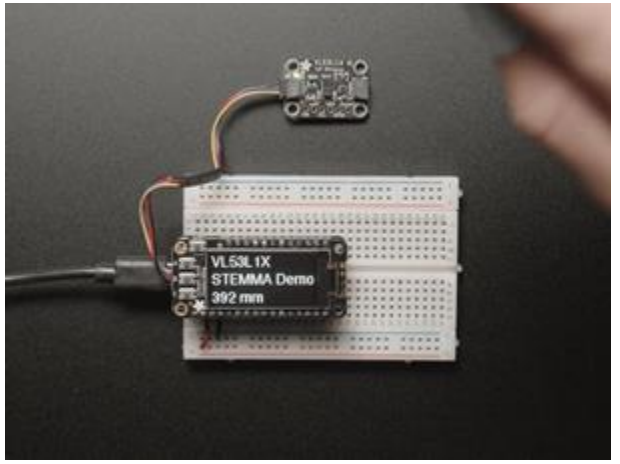
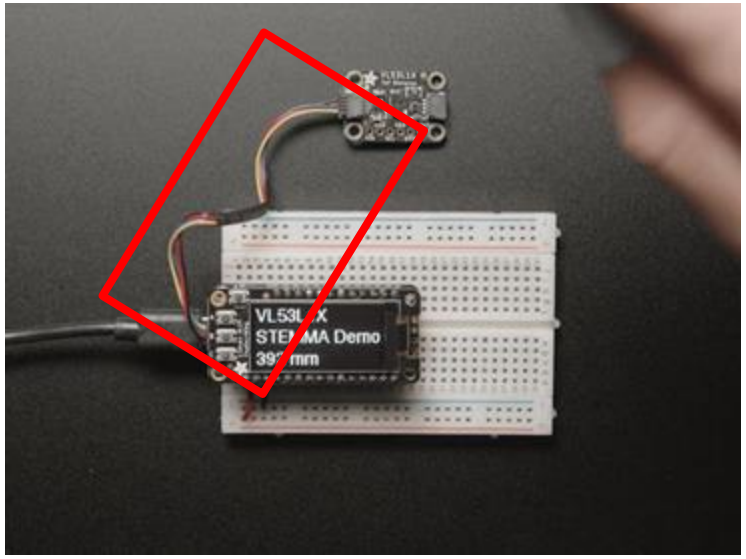


Illustration: https://www.adafruit.com

Digital sensors

"A type of sensor that directly outputs discrete digital data, typically in the form of binary signals (0s and 1s), representing the physical quantity it measures."

- …when data has been converted,
  - …it is then sent to a microcontroller over a digital communication protocol, such as



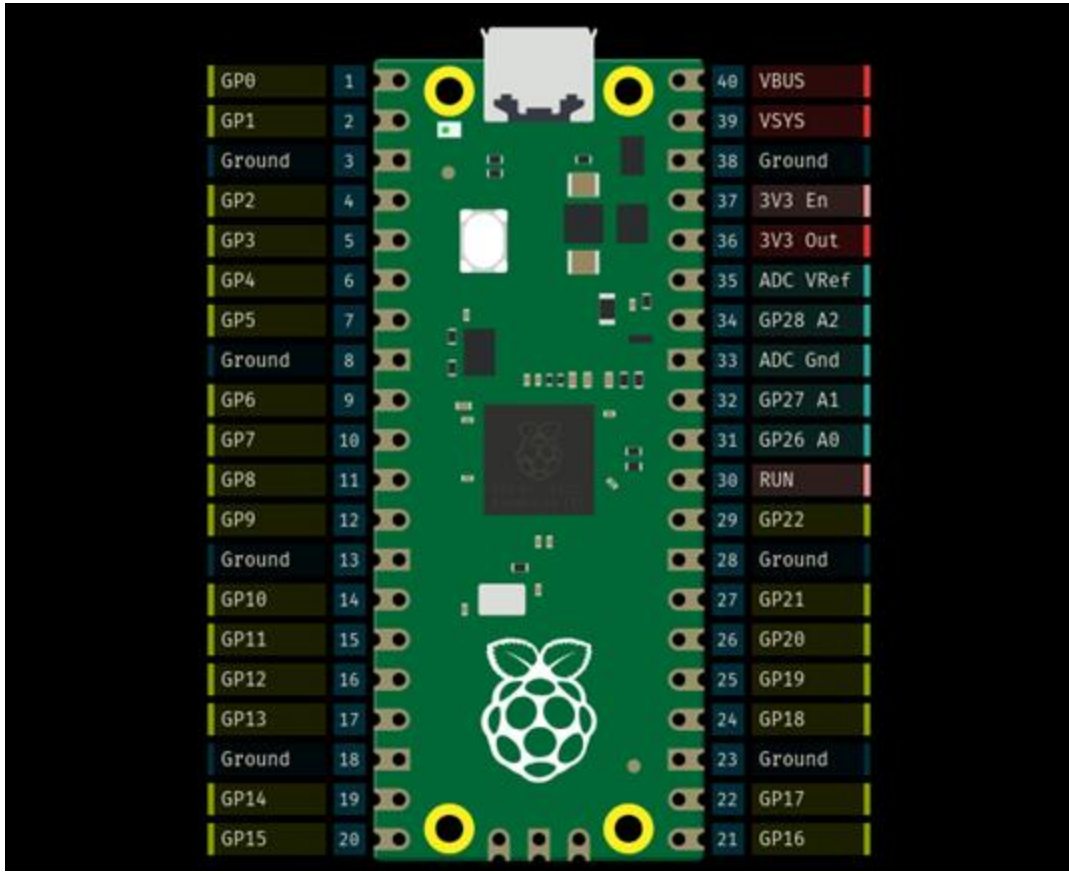


Illustration: https://pico.pinout.xyz/

Digital sensors

"A type of sensor that directly outputs discrete digital data, typically in the form of binary signals (0s and 1s), representing the physical quantity it measures."

- …when data has been converted,
  - …it is then sent to a microcontroller over a digital communication protocol, such as
    - **I2C**,

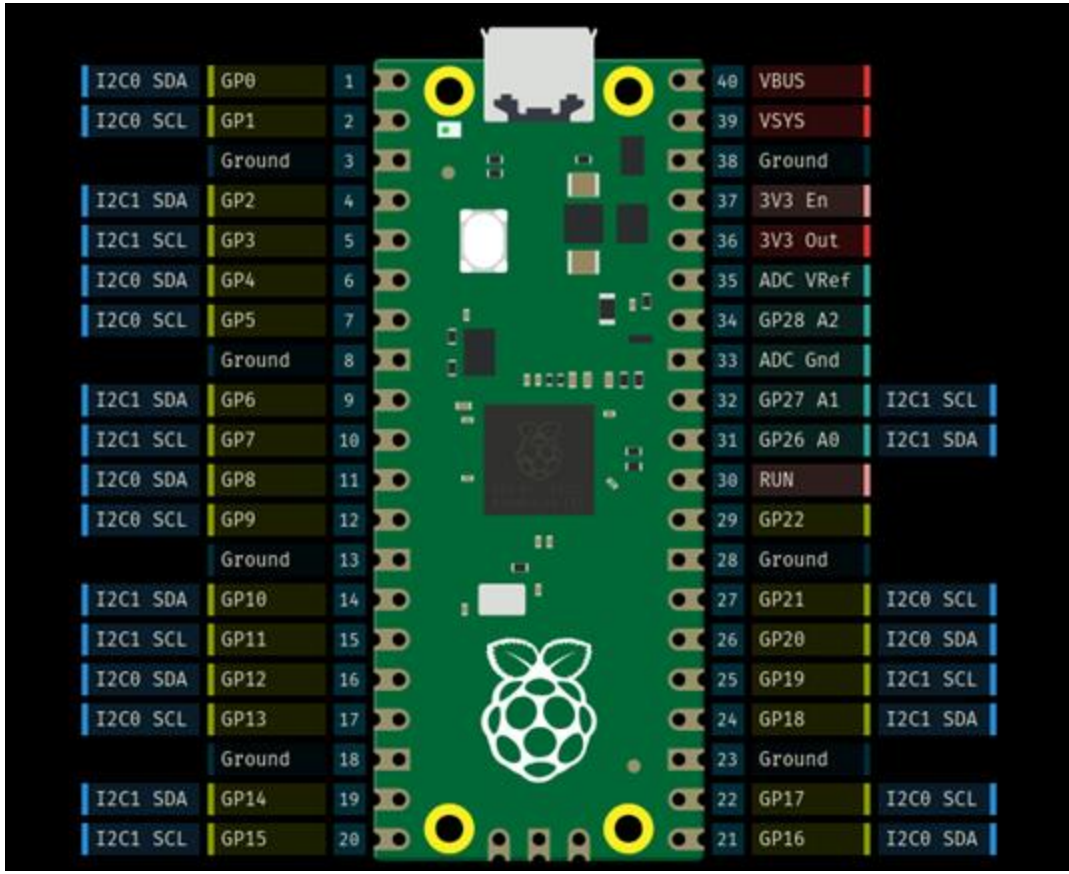


Illustration: https://pico.pinout.xyz/

Digital sensors

"A type of sensor that directly outputs discrete digital data, typically in the form of binary signals (0s and 1s), representing the physical quantity it measures."

- …when data has been converted,
  - …it is then sent to a microcontroller over a digital communication protocol, such as
    - I2C,
    - **SPI**, or

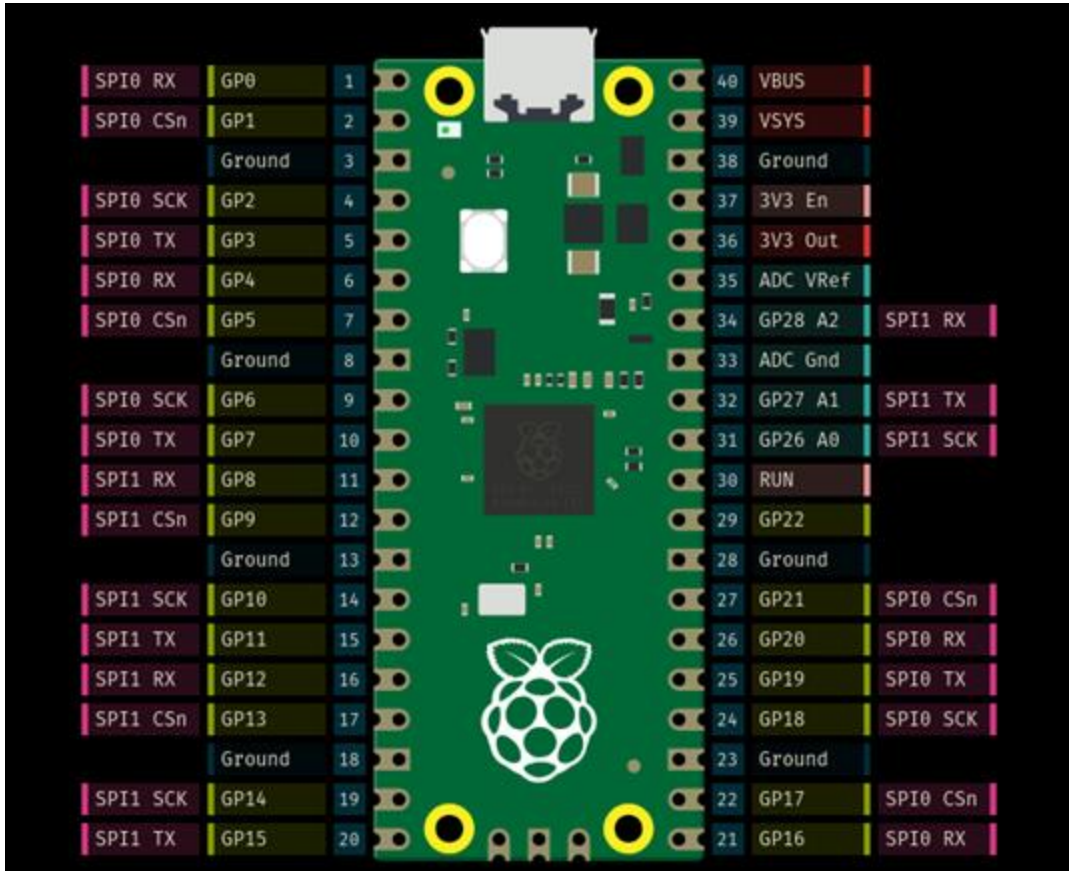


Illustration: https://pico.pinout.xyz/

Digital sensors

"A type of sensor that directly outputs discrete digital data, typically in the form of binary signals (0s and 1s), representing the physical quantity it measures."

- …when data has been converted,
  - …it is then sent to a microcontroller over a digital communication protocol, such as
    - I2C,
    - SPI, or
    - **UART**

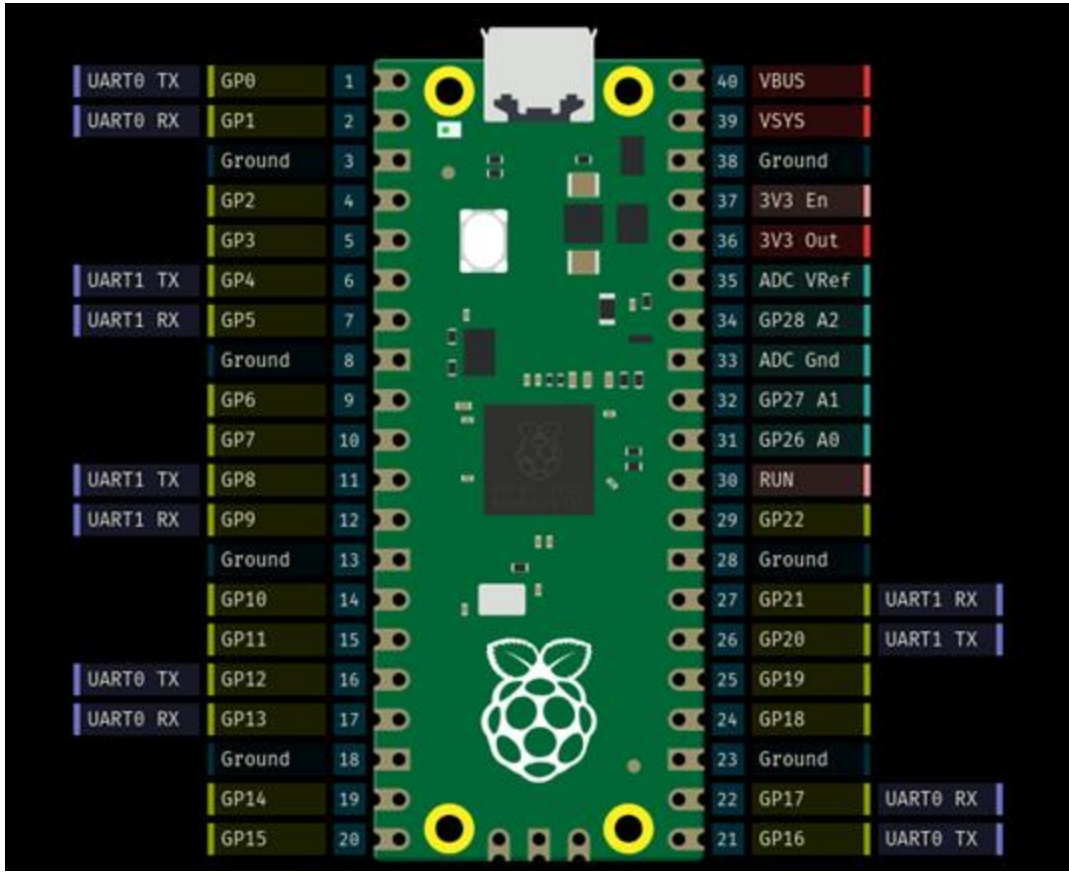


Illustration: https://pico.pinout.xyz/

Digital sensors

"A type of sensor that directly outputs discrete digital data, typically in the form of binary signals (0s and 1s), representing the physical quantity it measures."

- …when data has been converted,
  - …it is then sent to a microcontroller over a digital communication protocol, such as
    - I2C,
    - SPI, or
    - UART
  - **(more about that in module 9)**
    - **…but** a brief overview of the protocols wiring

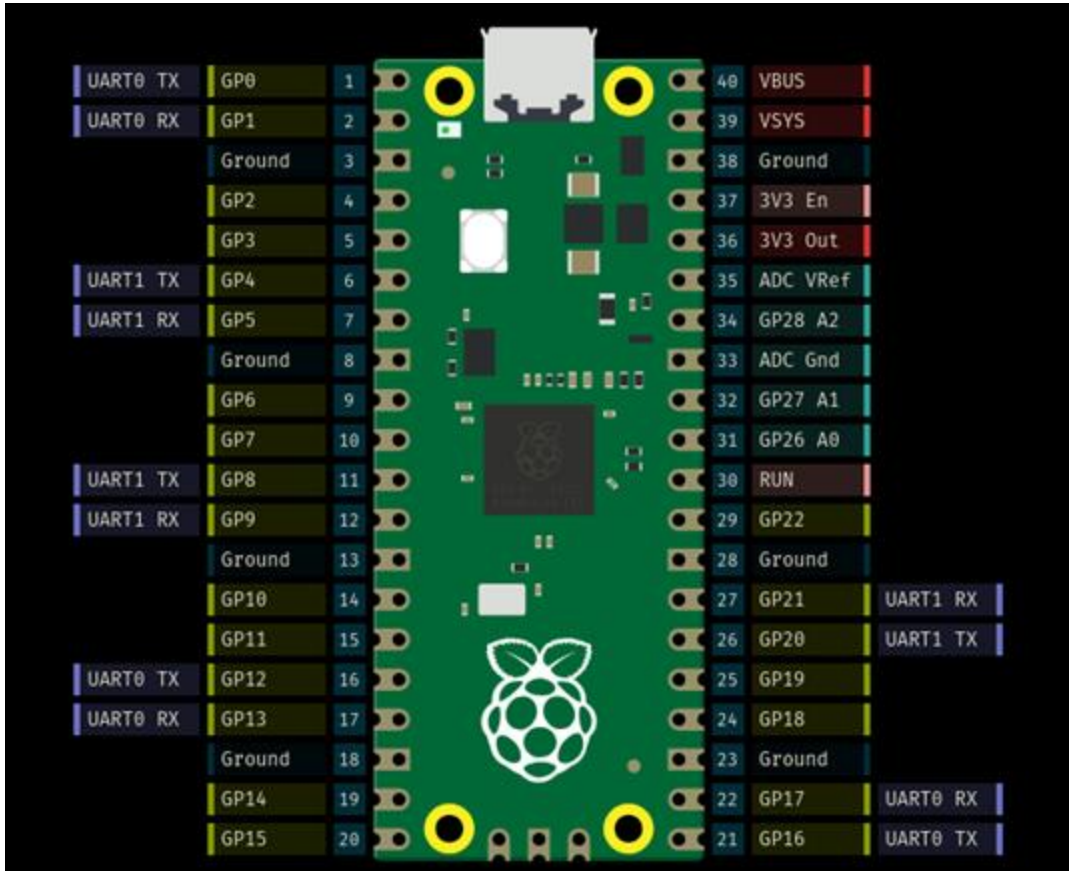


Illustration: https://pico.pinout.xyz/

Digital sensors

"A type of sensor that directly outputs discrete digital data, typically in the form of binary signals (0s and 1s), representing the physical quantity it measures."

- Connecting sensors using I2C (wiring)
  - **SDA** from the sensor to the microcontroller's **SDA** pin.
  - **SCL** from the sensor to the microcontroller's **SCL** pin.
  - **VCC** to the microcontroller's 3.3V or 5V (depending on the sensor).
  - **GND** to the microcontroller's ground (GND).

Two-Wire Interface, where:
  - **SDA**: Data line for transferring data
  - **SCL**: Clock line for synchronization

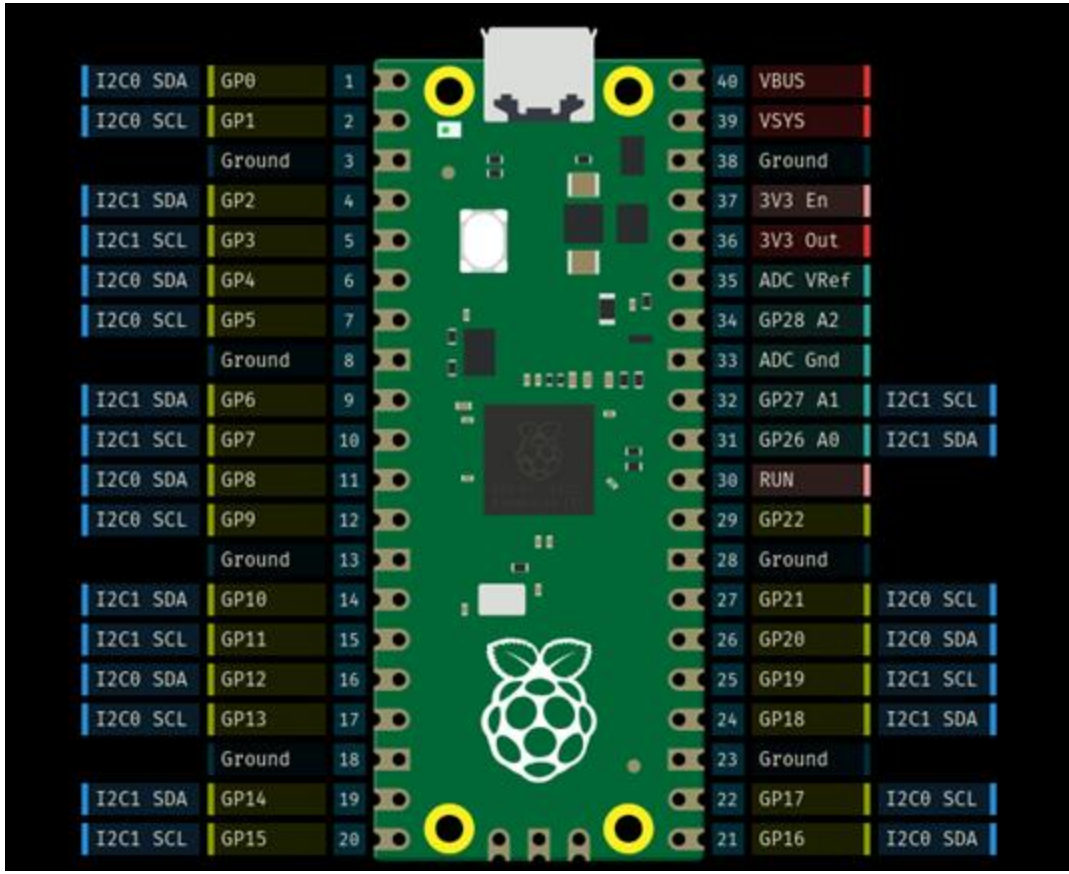


Illustration: https://pico.pinout.xyz/

Digital sensors

"A type of sensor that directly outputs discrete digital data, typically in the form of binary signals (0s and 1s), representing the physical quantity it measures."

- Connecting sensors using SPI (wiring)
  - **SCK** from the sensor to the microcontroller's **SCK** pin.
  - **MOSI** from the sensor to the microcontroller's **MOSI** pin.
  - **MISO** from the sensor to the microcontroller's **MISO** pin.
  - **CS** from the sensor to a general-purpose **CS** pin on the microcontroller.
  - **VCC** to the microcontroller's 3.3V or 5V.
  - **GND** to the microcontroller's ground (GND).

Four-Wire Interface, where**:**

- **SCK:** Clock line for synchronization
- **MOSI:** Master Out Slave In
  - (data sent from the microcontroller to the sensor)
- **MISO:** Master In Slave Out
  - (data sent from the sensor to the microcontroller)
- **CS:** Chip Select (used to select the sensor)

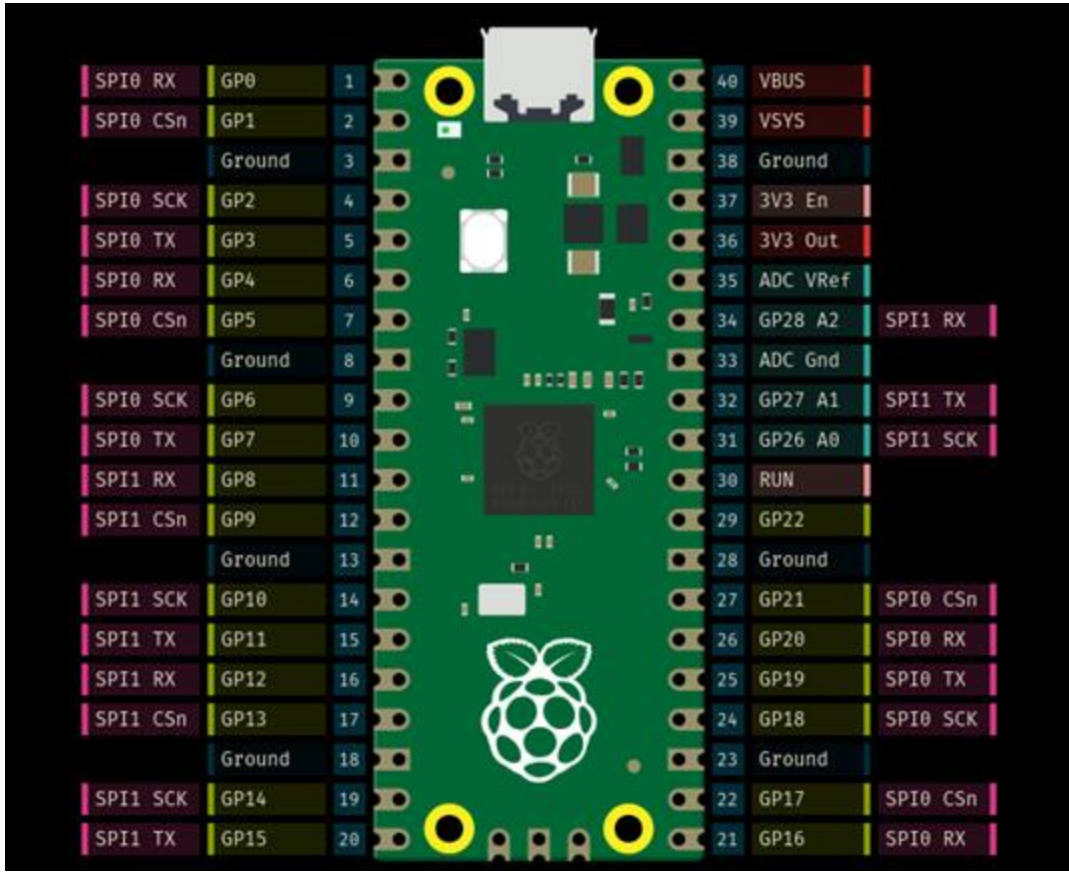


Illustration: https://pico.pinout.xyz/

Digital sensors

"A type of sensor that directly outputs discrete digital data, typically in the form of binary signals (0s and 1s), representing the physical quantity it measures."

- Connecting sensors using UART (wiring)
  - **TX** from the sensor to the microcontroller's **RX** pin. (counterpart)
  - **RX** from the sensor to the microcontroller's **TX** pin. (counterpart)
  - **VCC** to the microcontroller's 3.3V or 5V (depending on the sensor).
  - **GND** to the microcontroller's ground (GND).

Two-Wire Interface, where
  - **TX**: Transmit data from the microcontroller to the sensor
  - **RX**: Receive data from the sensor to the microcontroller

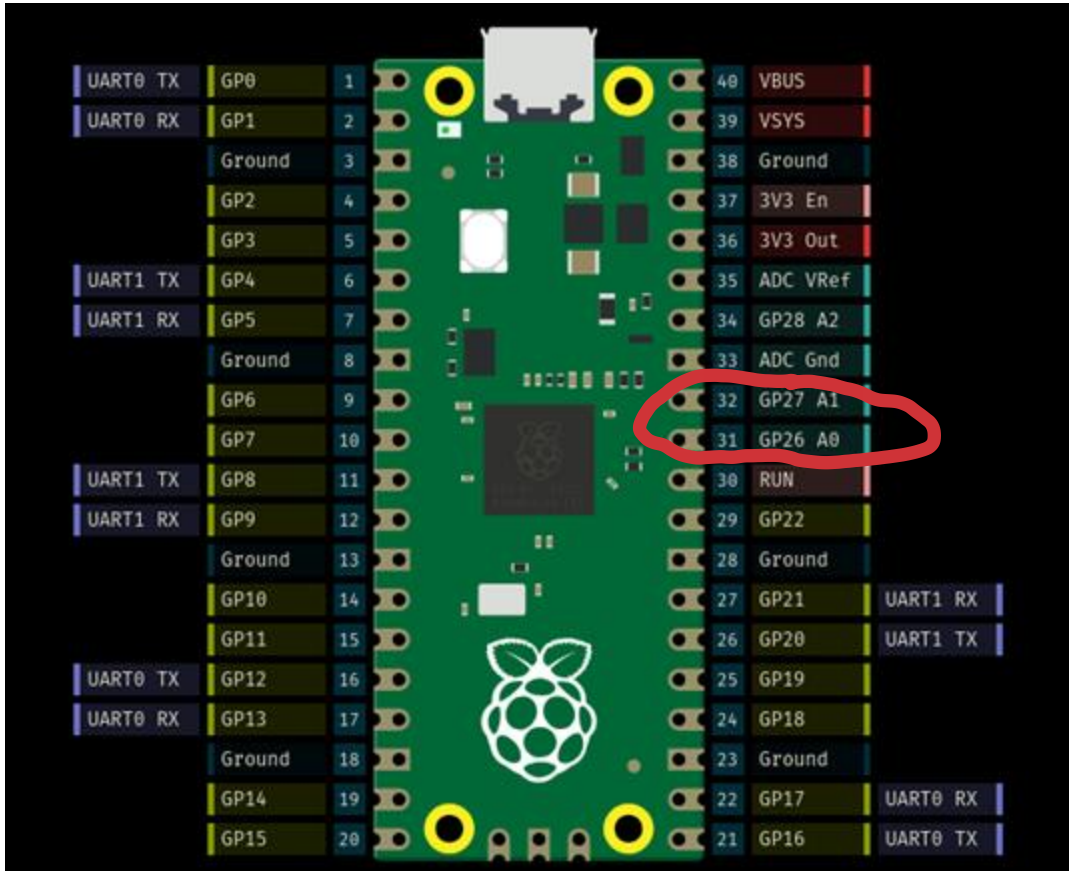
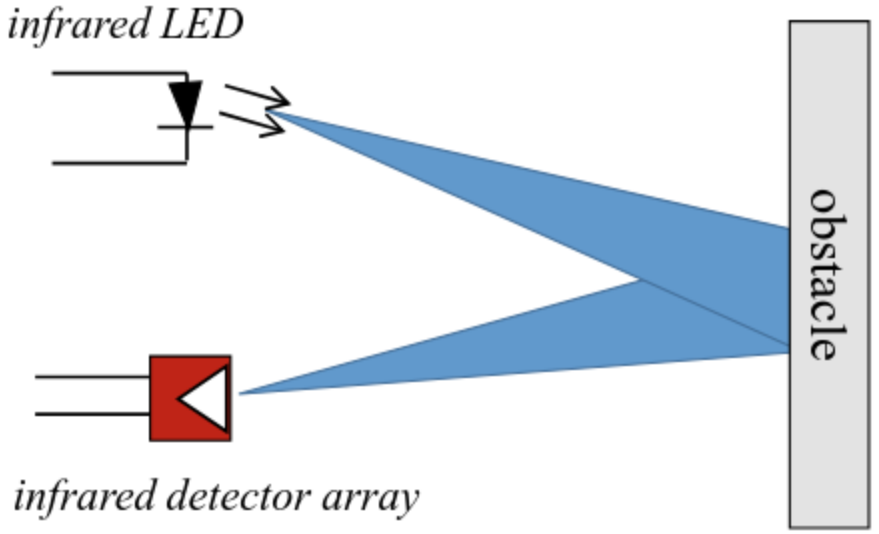


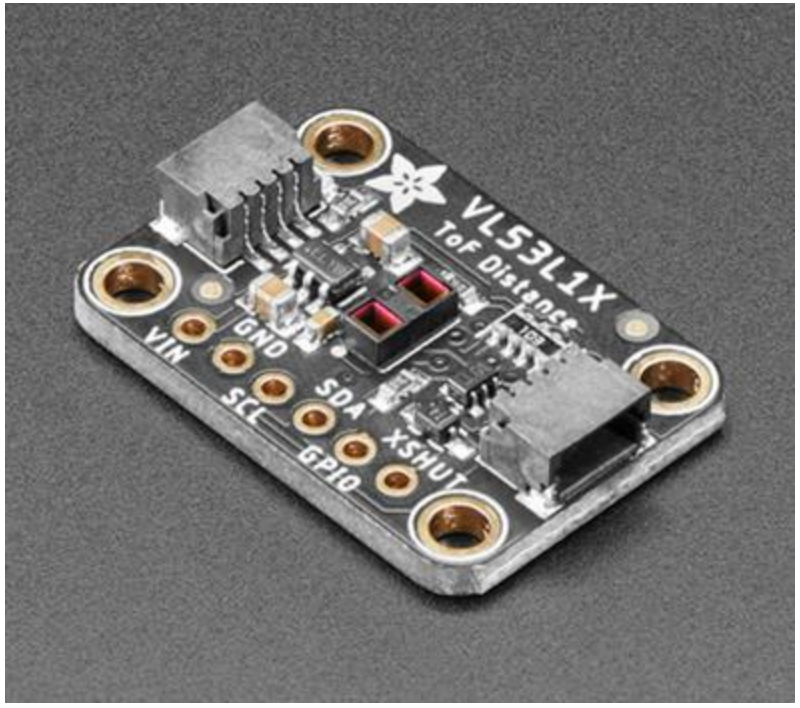
Illustration: https://pico.pinout.xyz/

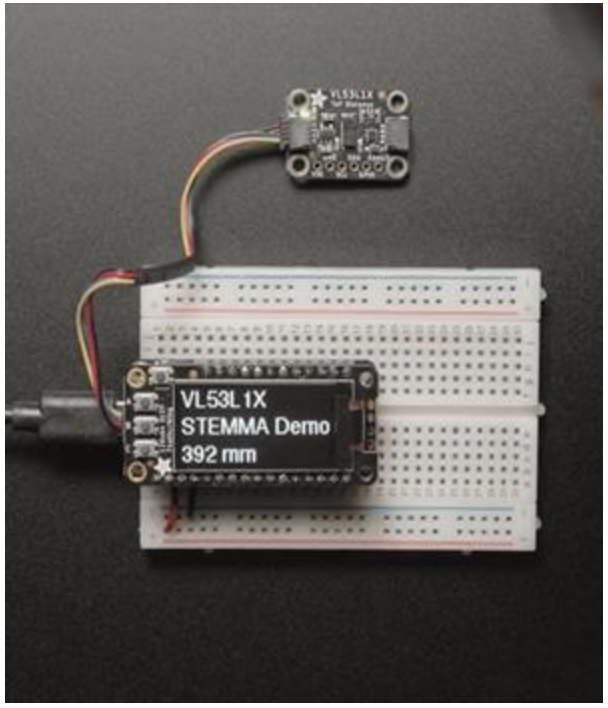
Digital sensors

"A type of sensor that directly outputs discrete digital data, typically in the form of binary signals (0s and 1s), representing the physical quantity it measures."

- **Example: Interface a VL53L1X**







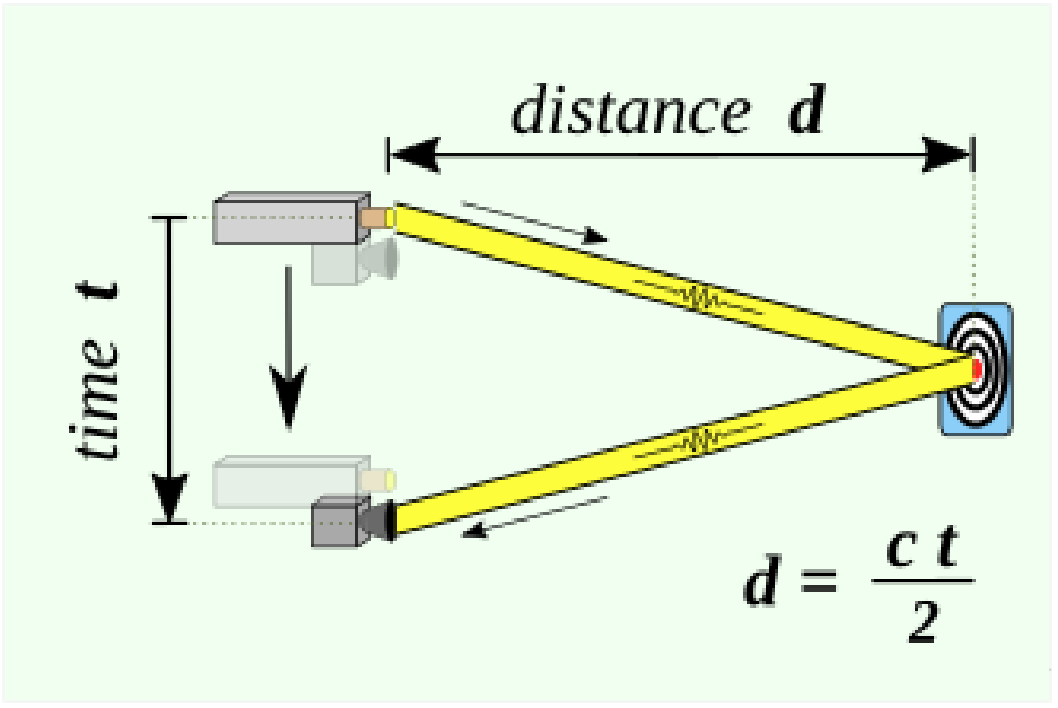


Illustration: https://www.adafruit.com

Digital sensors

"A type of sensor that directly outputs discrete digital data, typically in the form of binary signals (0s and 1s), representing the physical quantity it measures."

- **Example: Interface a VL53L1X**
  1. **Include the VL53L1X Library for MicroPython.**

Digital sensors

"A type of sensor that directly outputs discrete digital data, typically in the form of binary signals (0s and 1s), representing the physical quantity it measures."

- **Example: Interface a VL53L1X**
  1. **Include the VL53L1X Library for MicroPython.**

     Protocol: I²C interface (up to 400 kHz), Default address 0x29

     Since there is no built-in driver for VL53L1X, you will have to either:
     a. Write the library yourself based on the datasheet… or,

Figure 16. **Data format (sequential read)**

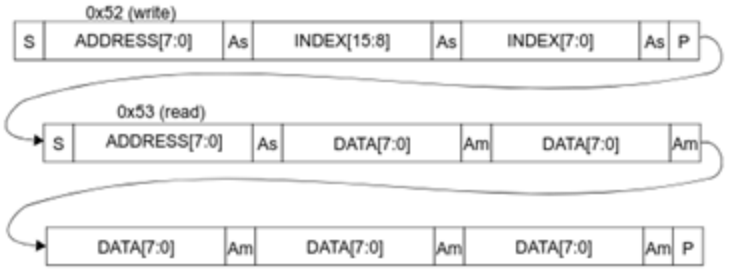


Figure 6. **Maximum distance and repeatability error vs. timing budget**

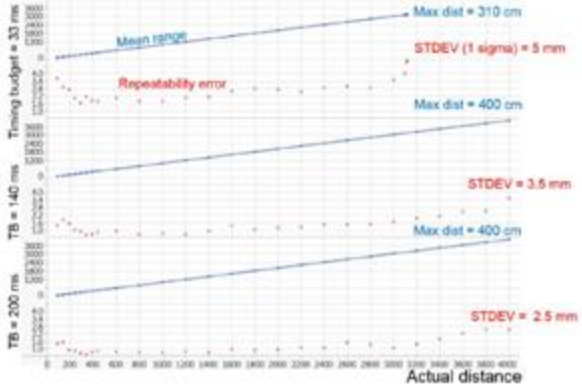
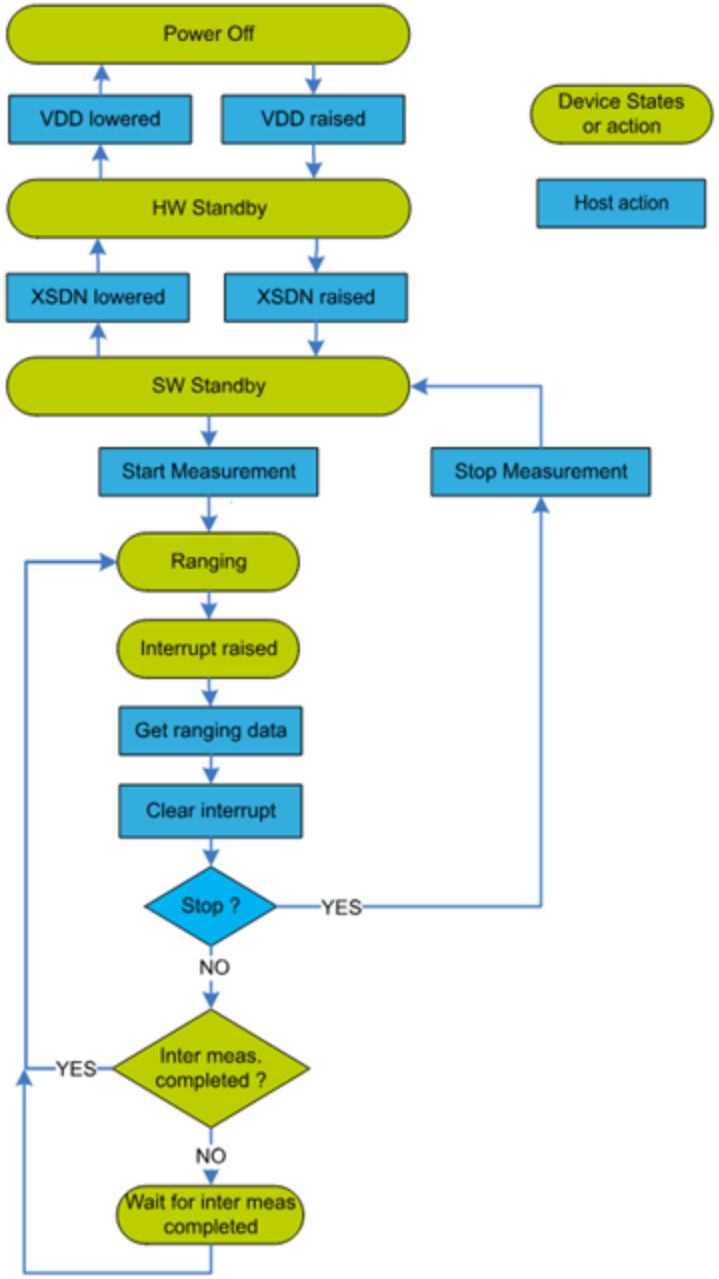


Figure 5. **System state machine**



Digital sensors

"A type of sensor that directly outputs discrete digital data, typically in the form of binary signals (0s and 1s), representing the physical quantity it measures."

- **Example: Interface a VL53L1X**
  1. **Include the VL53L1X Library for MicroPython.**

     Protocol: I²C interface (up to 400 kHz), Default address 0x29

     Since there is no built-in driver for VL53L1X, you will have to either:
     a. Write the library yourself based on the datasheet… or,

     b. Search for an existing implementation, eg.
        i. https://github.com/drakxtwo/vl53l1x_pico/blob/main/vl53l1x.py

     Upload the custom library, eg. named as `vl53l1x.py`, to your Pico.

Example: Interface a VL53L1X

```
# File: vl53l1x.py
import machine

VL51L1X_DEFAULT_CONFIGURATION = bytes([
0x00, # 0x2d : set bit 2 and 5 to 1 for fast plus mode
(1MHz I2C), else don't touch */
0x00, # 0x2e : bit 0 if I2C pulled up at 1.8V, else set
bit 0 to 1 (pull up at AVDD) */
0x00, # 0x2f : bit 0 if GPIO pulled up at 1.8V, else
set bit 0 to 1 (pull up at AVDD) */
0x01, # 0x30 : set bit 4 to 0 for active high interrupt
and 1 for active low (bits 3:0 must be 0x1), use
SetInterruptPolarity() */
0x02, # 0x31 : bit 1 = interrupt depending on the
polarity, use CheckForDataReady() */
0x00, # 0x32 : not user-modifiable */
0x02, # 0x33 : not user-modifiable */
0x08, # 0x34 : not user-modifiable */
0x00, # 0x35 : not user-modifiable */
0x08, # 0x36 : not user-modifiable */
0x10, # 0x37 : not user-modifiable */
0x01, # 0x38 : not user-modifiable */
0x01, # 0x39 : not user-modifiable */
...
```

Link: https://github.com/drakxtwo/vl53l1x_pico/blob/main/vl53l1x.py

Digital sensors

"A type of sensor that directly outputs discrete digital data, typically in the form of binary signals (0s and 1s), representing the physical quantity it measures."

- **Example: Interface a VL53L1X**
  1. Include the VL53L1X Library for MicroPython.
  2. **Setup circuit, such that the I2C connections are correct**
     a. Either using **pins headers**, or

Man kan enten lodde det sammen med pico'en

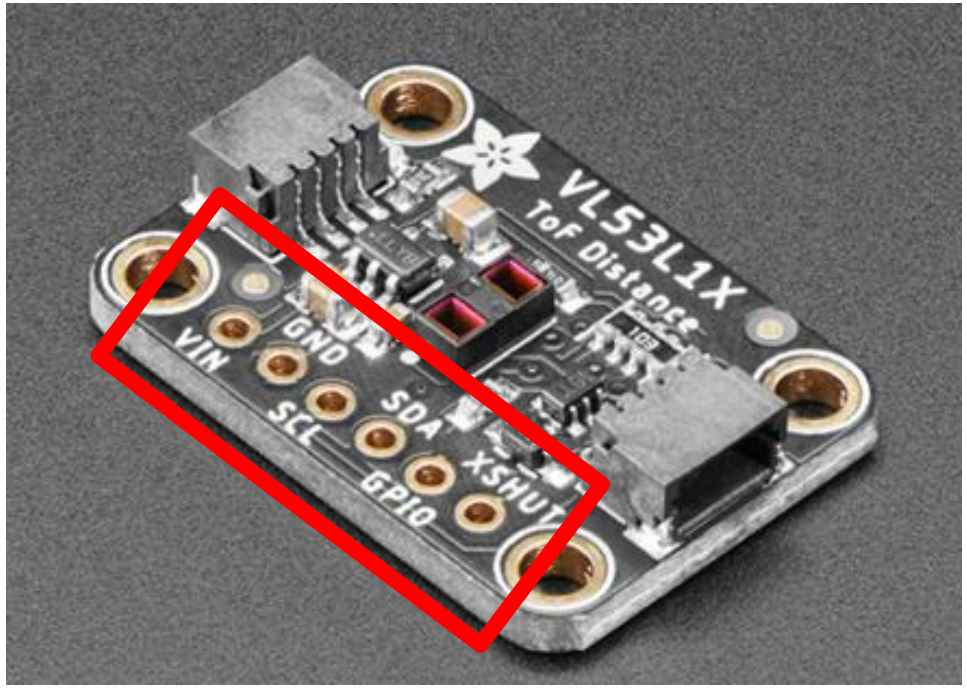


Illustration: https://www.adafruit.com

Digital sensors

Eller bruge kabel

"A type of sensor that directly outputs discrete digital data, typically in the form of binary signals (0s and 1s), representing the physical quantity it measures."

- **Example: Interface a VL53L1X**
  1. Include the VL53L1X Library for MicroPython.

  2. **Setup circuit, such that the I2C connections are correct**
     a. Either using **pins headers**, or
     b. STEMMA QT connector

Så kan man data chain dem

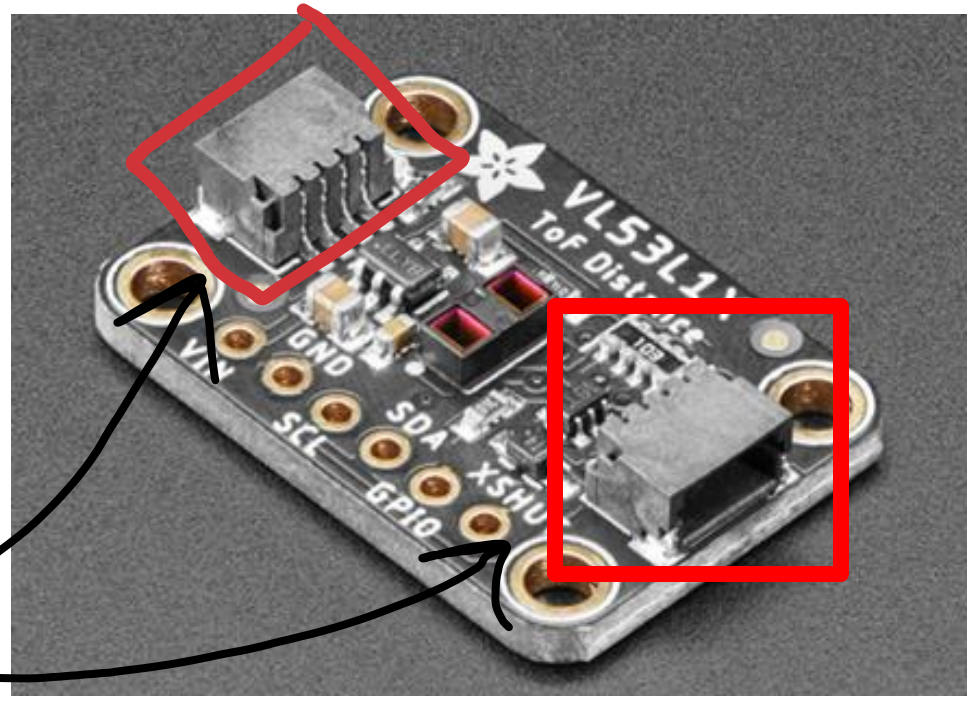


Illustration: https://www.adafruit.com

Digital sensors

"A type of sensor that directly outputs discrete digital data, typically in the form of binary signals (0s and 1s), representing the physical quantity it measures."

- **Example: Interface a VL53L1X**
  1. Include the VL53L1X Library for MicroPython.
  2. Setup circuit, such that the I2C connections are correct
  3. **Interface the VL53L1X by including the custom library**

Example: Interface a VL53L1X

```
# File: main.py
import machine
import time

# Include the custom libary
from vl53l1x import VL53L1X

# Initialize I2C - Adjust pins depending on your
circuit setup
i2c_dev = machine.I2C(1, sda=machine.Pin(26),
scl=machine.Pin(27))

# Initialize the VL53L1X sensor
distance = VL53L1X(i2c_dev)

while True:
    print("range: mm ", distance.read())

    # Wait for 100 ms before the next reading
    time.sleep_ms(100)
```

Digital sensors

"A type of sensor that directly outputs discrete digital data, typically in the form of binary signals (0s and 1s), representing the physical quantity it measures."

- **Example: Interface a VL53L1X**
  1. Include the VL53L1X Library for MicroPython.
  2. Setup circuit, such that the I2C connections are correct
  3. Interface the VL53L1X by including the custom library

Example: Interface a VL53L1X

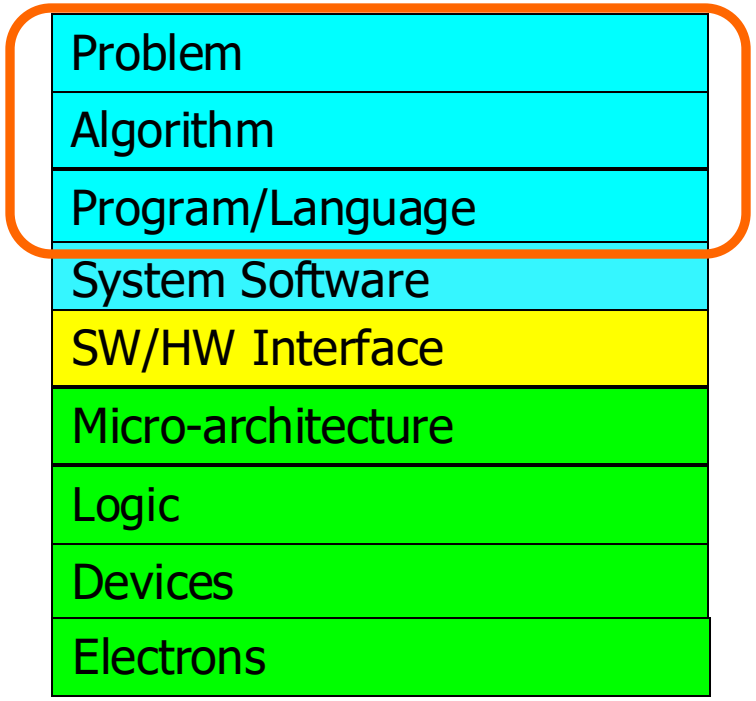
```
# File: main.py
import machine
import time

# Include the custom libary
from vl53l1x import VL53L1X

# Initialize I2C - Adjust pins depending on your
circuit setup
i2c_dev = machine.I2C(1, sda=machine.Pin(26),
scl=machine.Pin(27))

# Initialize the VL53L1X sensor
distance = VL53L1X(i2c_dev)

while True:
    print("range: mm ", distance.read())

    # Wait for 100 ms before the next reading
    time.sleep_ms(100)
```

Demo?

Robot Behaviors

Robot Behaviors

"The actions or responses performed by a robot based on the information it receives from its environment through sensors and its programmed instructions."



Robot Behaviors

"The actions or responses performed by a robot based on the information it receives from its environment through sensors and its programmed instructions."

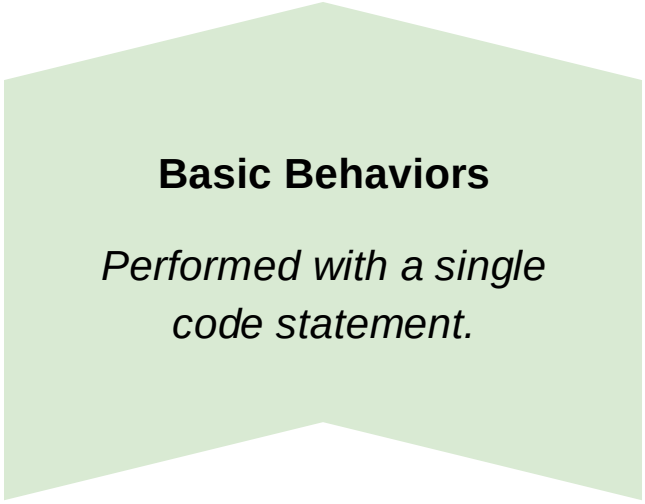
- **Levels of Robot Behaviors**

Reference: https://docs.idew.org/code-robotics/references/robot-behaviors

Robot Behaviors

"The actions or responses performed by a robot based on the information it receives from its environment through sensors and its programmed instructions."

- **Levels of Robot Behaviors**
  - **Basic Behaviors** – low-level behaviors that perform a single action, such as: turning on the motors, reading a sensor, etc.

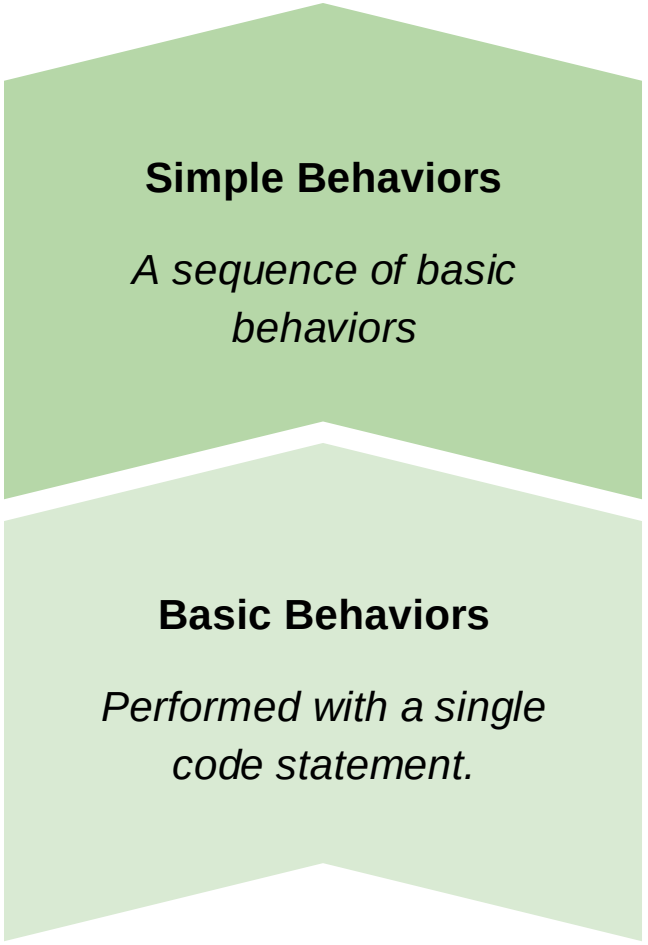


Reference: https://docs.idew.org/code-robotics/references/robot-behaviors

Robot Behaviors

"The actions or responses performed by a robot based on the information it receives from its environment through sensors and its programmed instructions."

- **Levels of Robot Behaviors**
  - **Basic Behaviors** – low-level behaviors that perform a single action, such as: turning on the motors, reading a sensor, etc.
  - **Simple Behaviors** – mid-level behaviors that perform a simple task, such as: driving forward for 5 seconds, turning to the right, etc.

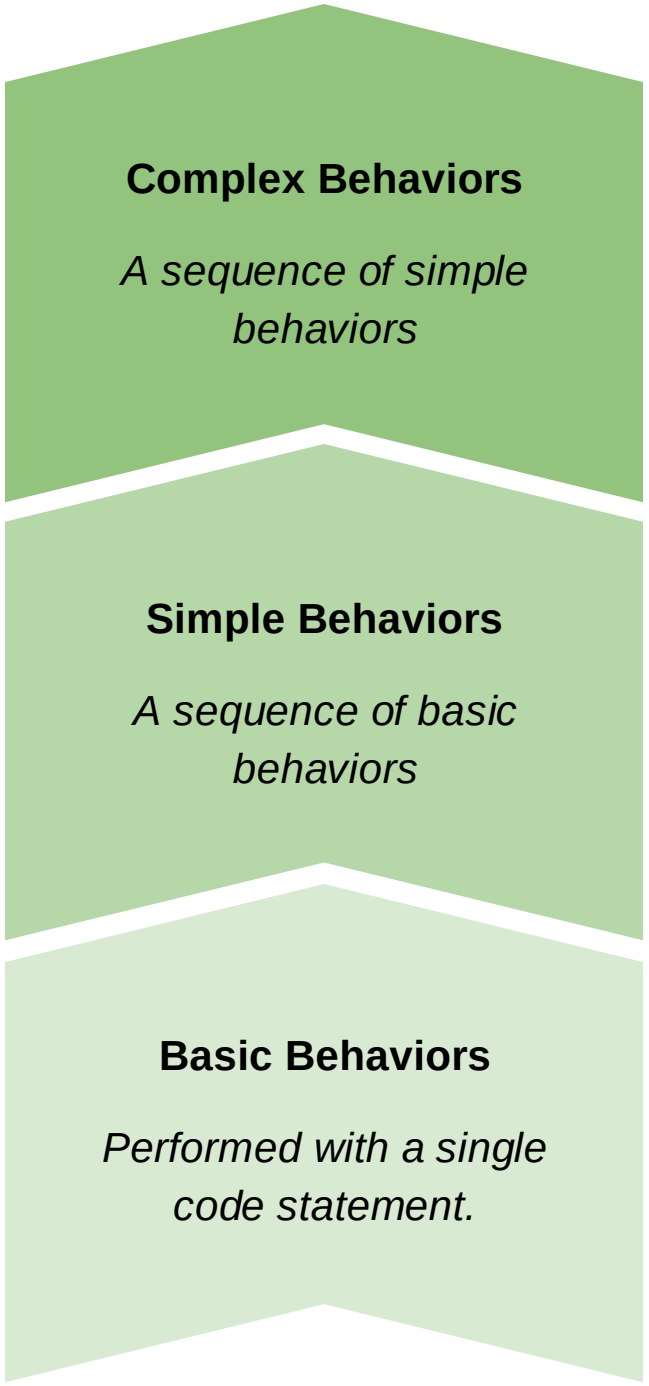


Reference: https://docs.idew.org/code-robotics/references/robot-behaviors

Robot Behaviors

"The actions or responses performed by a robot based on the information it receives from its environment through sensors and its programmed instructions."

- **Levels of Robot Behaviors**
  - **Basic Behaviors** – low-level behaviors that perform a single action, such as: turning on the motors, reading a sensor, etc.
  - **Simple Behaviors** – mid-level behaviors that perform a simple task, such as: driving forward for 5 seconds, turning to the right, etc.
  - **Complex Behaviors** – high-level behaviors that perform a complex task, such as: following a line, driving around an obstacle, etc.

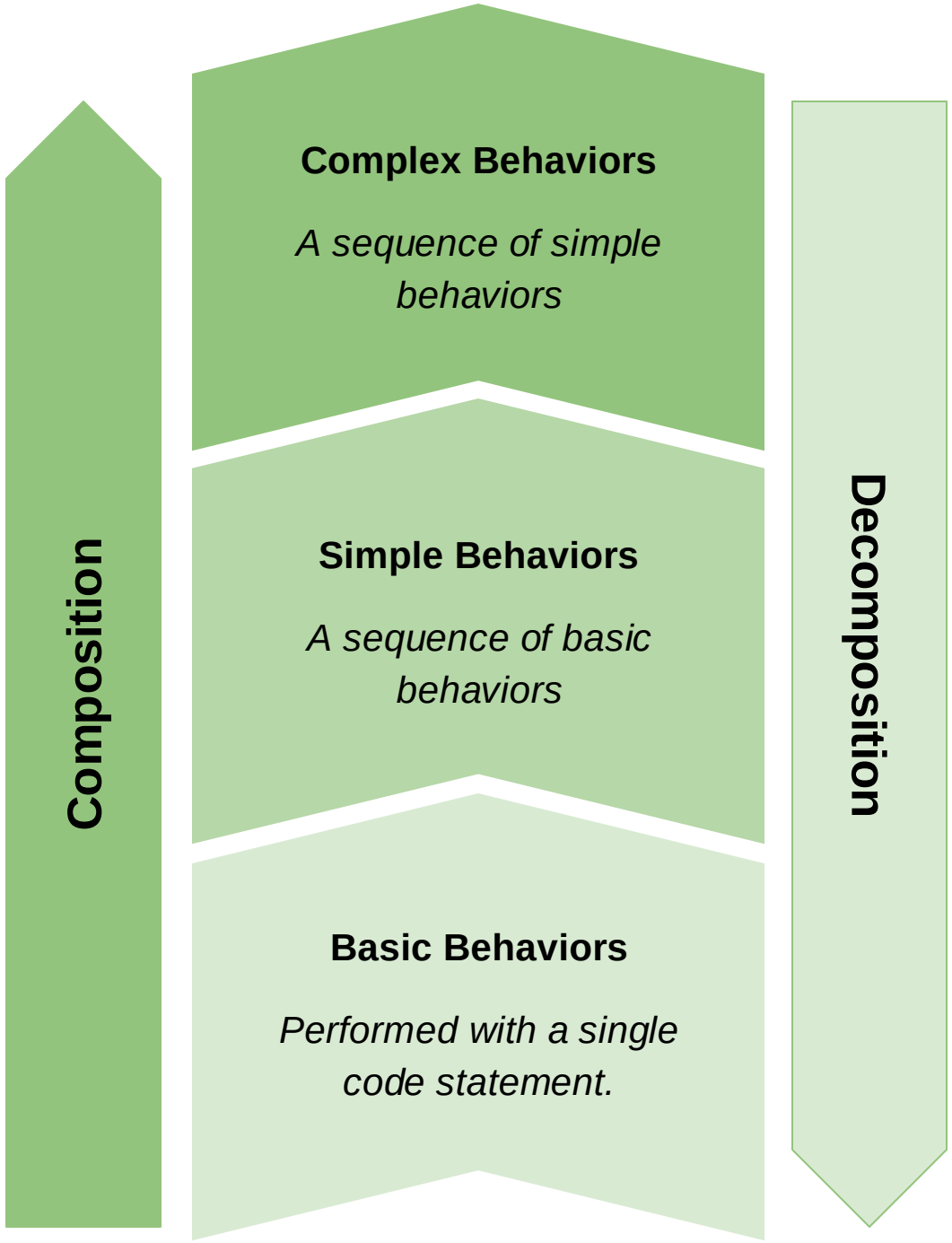


Reference: https://docs.idew.org/code-robotics/references/robot-behaviors

Robot Behaviors

"The actions or responses performed by a robot based on the information it receives from its environment through sensors and its programmed instructions."

- **Levels of Robot Behaviors**
  - **Composition:** Lower-level behaviors can be combined into a sequence that produces a more complex behavior.
  - **Decomposition**: A higher-level behavior can be broken down into a sequence of more simple behaviors.

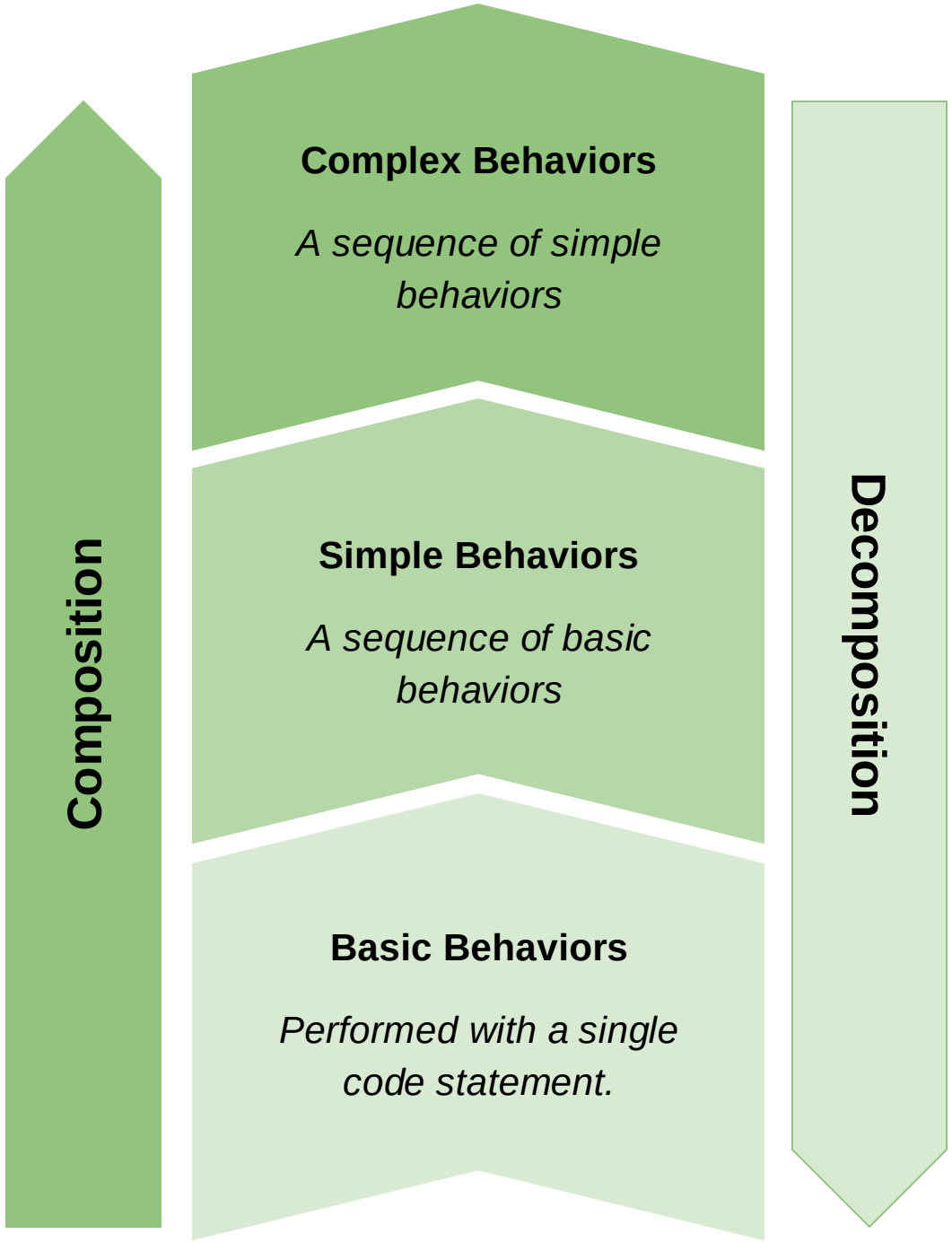


Reference: https://docs.idew.org/code-robotics/references/robot-behaviors

Robot Behaviors

"The actions or responses performed by a robot based on the information it receives from its environment through sensors and its programmed instructions."

- **How to define Robot Behaviors?**



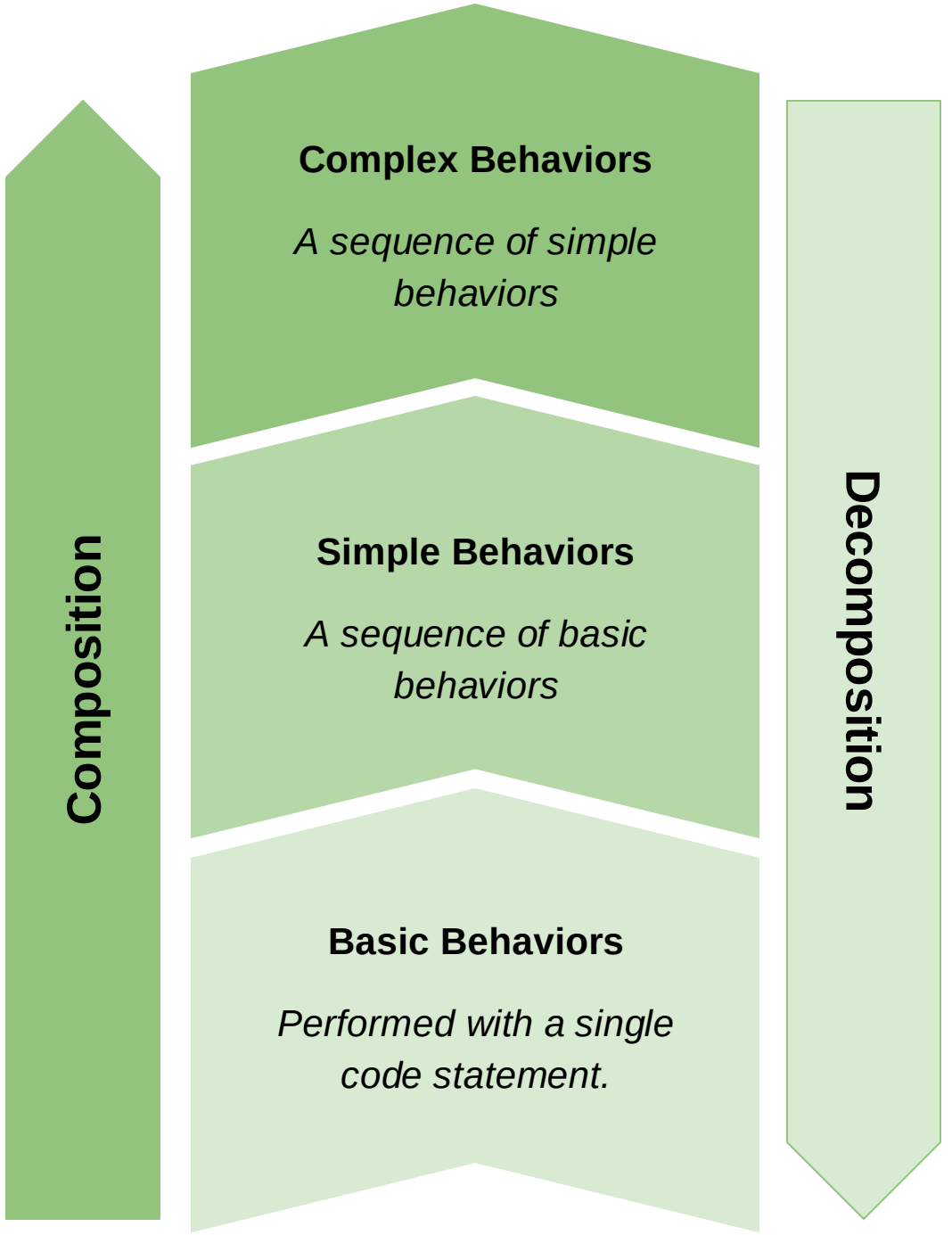
(Typically used to design and manage the behavior of mobile robots)

Reference: https://docs.idew.org/code-robotics/references/robot-behaviors

Robot Behaviors

"The actions or responses performed by a robot based on the information it receives from its environment through sensors and its programmed instructions."

- **How to define Robot Behaviors?**
  - **Pseudocode:** A high-level description of the steps needed to perform a task or solve a problem.



(Typically used to design and manage the behavior of mobile robots)

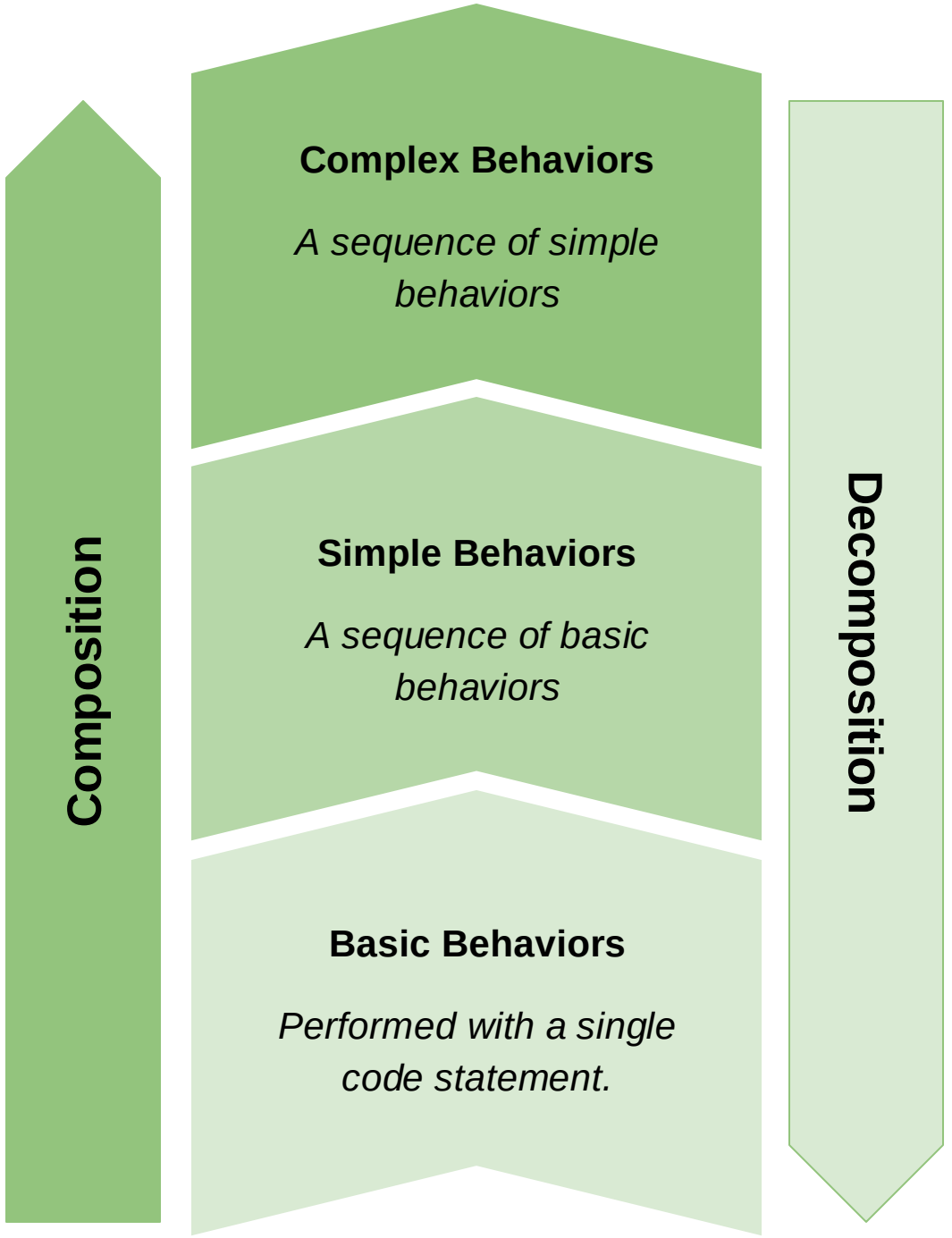
Reference: https://docs.idew.org/code-robotics/references/robot-behaviors

Robot Behaviors

"The actions or responses performed by a robot based on the information it receives from its environment through sensors and its programmed instructions."

- **How to define Robot Behaviors?**
  - **Pseudocode:** A high-level description of the steps needed to perform a task or solve a problem.
  - **Finite-State Machines:** A mathematical model of computation used to design systems that can be in one of a finite number of states at any given time.

(Typically used to design and manage the behavior of mobile robots)



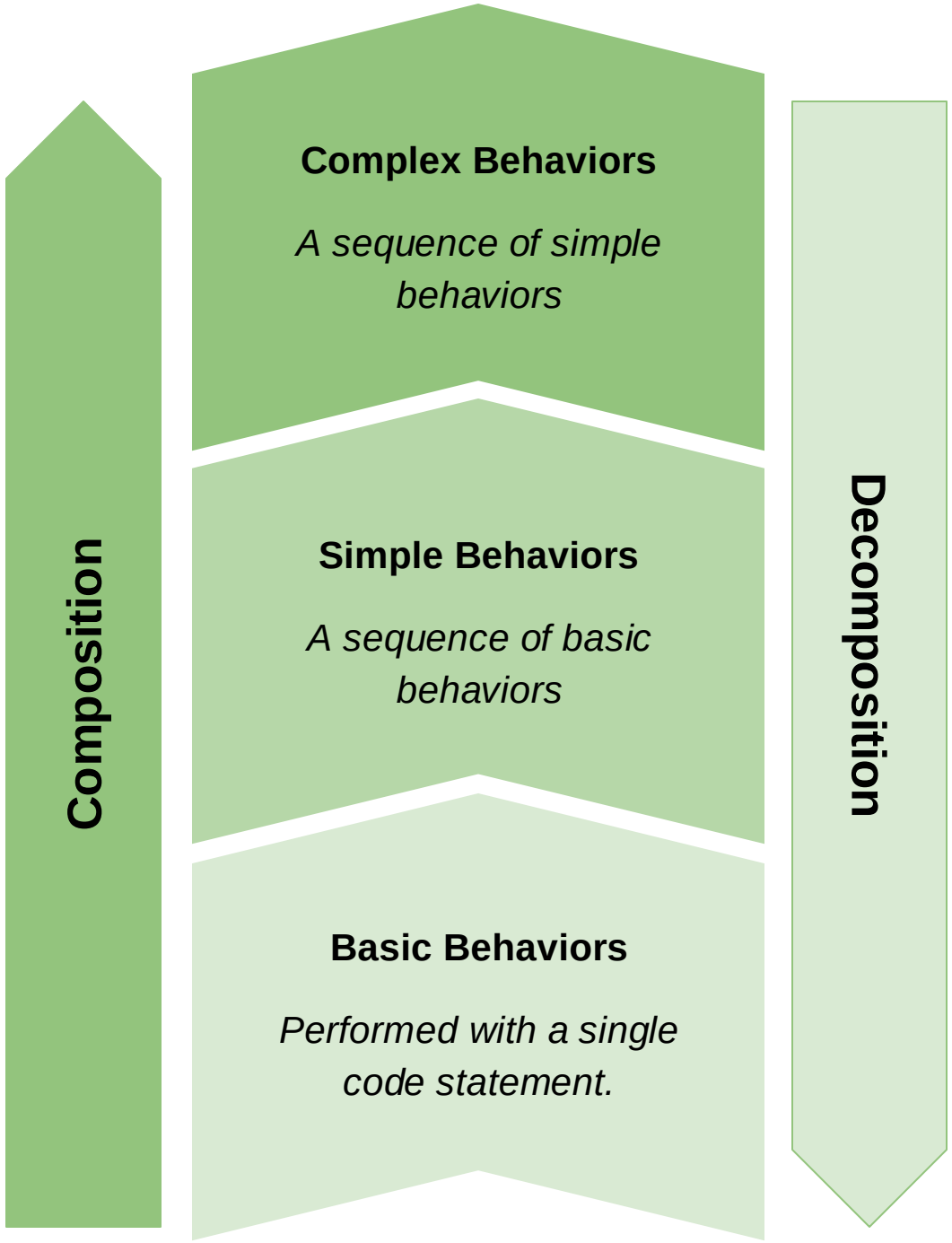
Reference: https://docs.idew.org/code-robotics/references/robot-behaviors

Robot Behaviors

"The actions or responses performed by a robot based on the information it receives from its environment through sensors and its programmed instructions."

- **How to define Robot Behaviors?**
  - **Pseudocode:** A high-level description of the steps needed to perform a task or solve a problem.
  - **Finite-State Machines:** A mathematical model of computation used to design systems that can be in one of a finite number of states at any given time.
  - **Flowcharts:** A visual representation of the logic flow in a robot's decision-making process, eg. in a specific state.

(Typically used to design and manage the behavior of mobile robots)



Reference: https://docs.idew.org/code-robotics/references/robot-behaviors

Pseudocode

Pseudocode

"...a high-level description of the steps needed to perform a task or solve a problem."

= describing the steps in an algorithm or a program.

Pseudocode for finding the largest number in a list

```
INITIALIZE a list of values
INITIALIZE a variable 'max' with the value of the first
element in the list

FOR each element in the list
    IF its value is greater than 'max'
        SET 'max' to its value
    END IF
END FOR

PRINT 'max' as the largest number
```

Converting the pseudocode to Python

```
list = [3, 5, 2, 8, 6]
max = list[0]

for num in list:
   if num > max:
       max = num

print("The largest number is:", max)
```

Example: Find the largest number in a list of numbers.

Pseudocode

"...a high-level description of the steps needed to perform a task or solve a problem."

= describing the steps in an algorithm or a program.

- **Characteristics**
  - **Simplified/Readable**: It should be easy to understand, with clear and concise language.
  - **Language-Independent:** It doesn't adhere to any particular programming language syntax.
  - **Descriptive:** Uses simple terms to describe operations, such as "IF," "THEN," "FOR," "WHILE," etc.
  - **Structured**: Though informal, it often follows common programming structures like loops, conditionals, and sequences.

Pseudocode for finding the largest number in a list

```
INITIALIZE a list of values
INITIALIZE a variable 'max' with the value of the first
element in the list

FOR each element in the list
    IF its value is greater than 'max'
        SET 'max' to its value
    END IF
END FOR

PRINT 'max' as the largest number
```

Converting the pseudocode to Python

```python
list = [3, 5, 2, 8, 6]
max = list[0]

for num in list:
   if num > max:
       max = num

print("The largest number is:", max)
```

Example: Find the largest number in a list of numbers.

State Machines

Finite-State Machines

*"A mathematical model of computation used to design systems that can be in one of a **finite** number of states at any given time."*

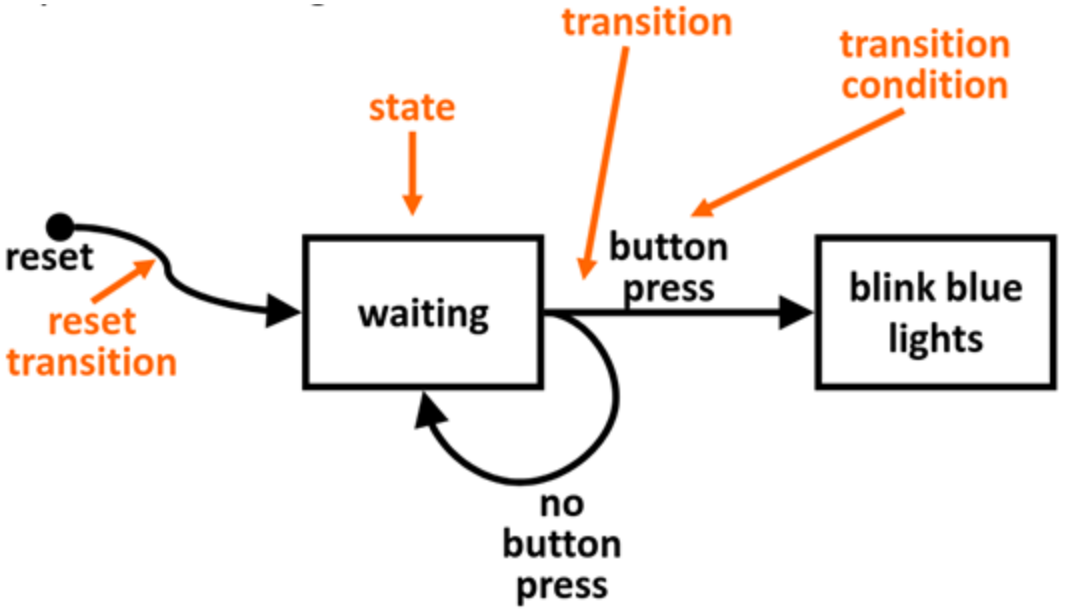


Illustration: https://www.clear.rice.edu/engi128/Handouts/Lec17-Robotics.pdf
https://en.wikipedia.org/wiki/Finite-state_machine

Finite-State Machines

*"A mathematical model of computation used to design systems that can be in one of a **finite** number of states at any given time."*

- **Components** (basic)
  - **States:** Distinct conditions or modes the robot can be in. Each state represents a specific action or set of actions.
    - **Actions:** Operations or behaviors executed when the robot is in a particular state.

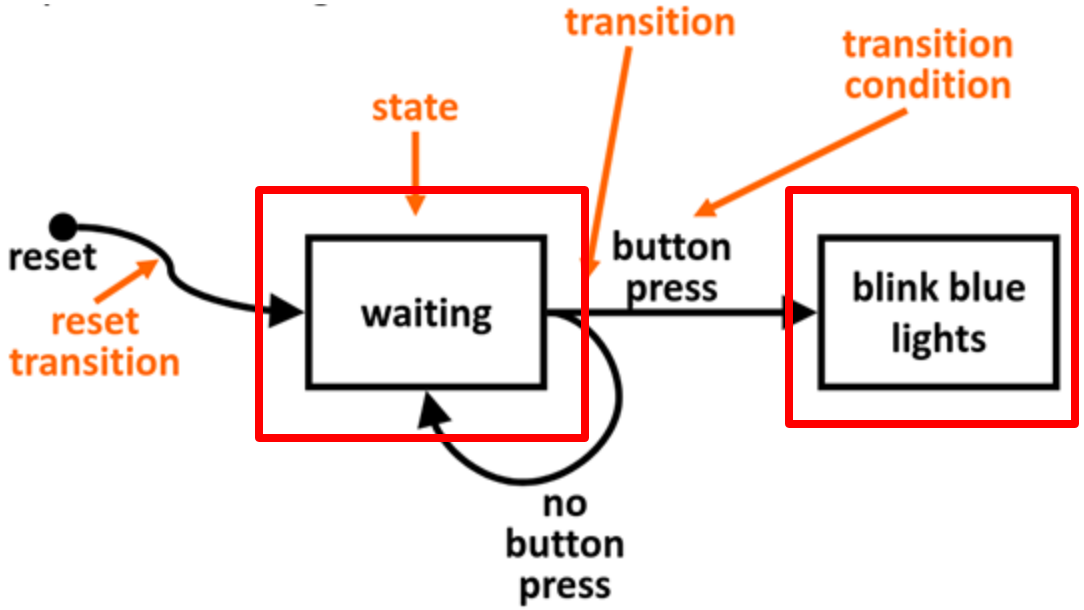


Illustration: https://www.clear.rice.edu/engi128/Handouts/Lec17-Robotics.pdf
https://en.wikipedia.org/wiki/Finite-state_machine

Finite-State Machines

*"A mathematical model of computation used to design systems that can be in one of a **finite** number of states at any given time."*

- **Components** (basic)
  - **States:** Distinct conditions or modes the robot can be in. Each state represents a specific action or set of actions.
  - **Transitions:** Rules or conditions that dictate when and how the robot moves from one state to another.

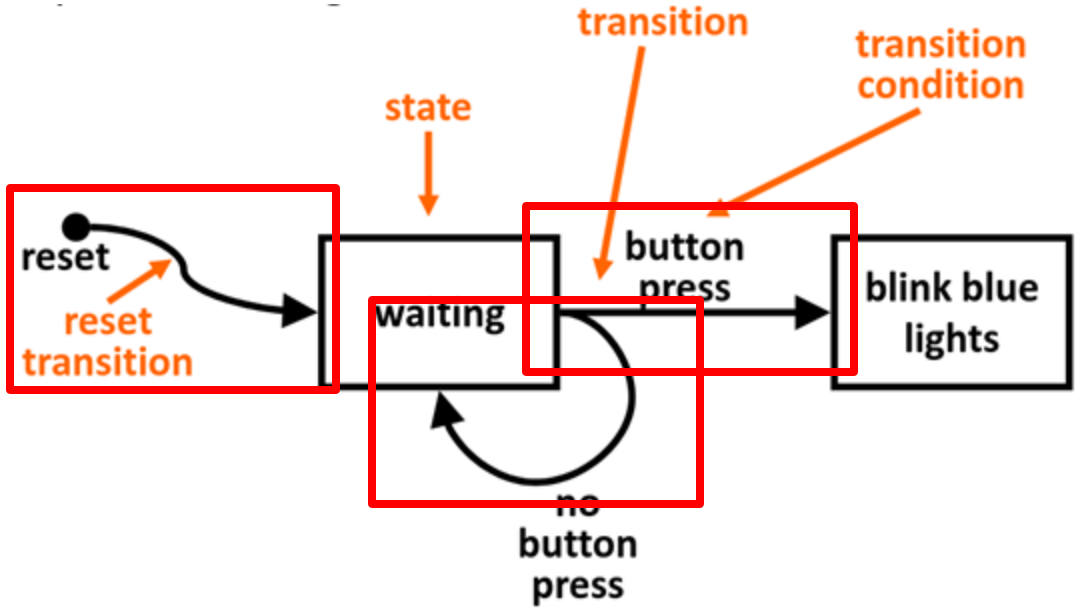


Illustration: https://www.clear.rice.edu/engi128/Handouts/Lec17-Robotics.pdf
https://en.wikipedia.org/wiki/Finite-state_machine

Finite-State Machines

*"A mathematical model of computation used to design systems that can be in one of a **finite** number of states at any given time."*

- **Components** (basic)
  - **States:** Distinct conditions or modes the robot can be in. Each state represents a specific action or set of actions.
  - **Transitions:** Rules or conditions that dictate when and how the robot moves from one state to another.
  - **Transition conditions:** External or internal signals (input/events) that trigger transitions between states.

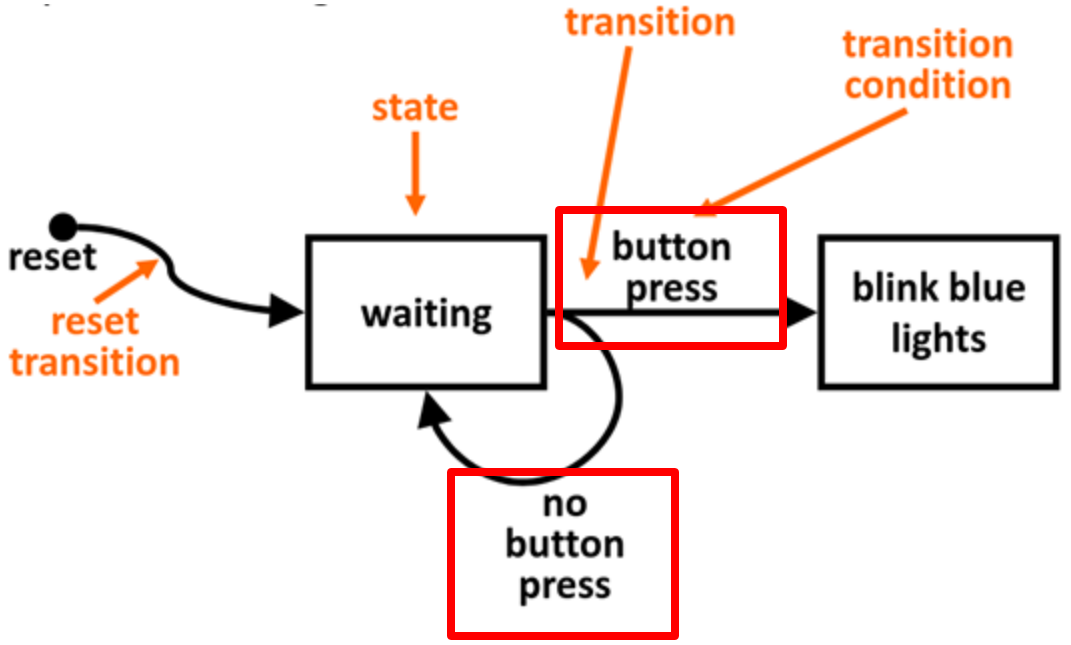


Illustration: https://www.clear.rice.edu/engi128/Handouts/Lec17-Robotics.pdf
https://en.wikipedia.org/wiki/Finite-state_machine

Finite-State Machines

*"A mathematical model of computation used to design systems that can be in one of a **finite** number of states at any given time."*

- **Components** (basic)
  - **States:** Distinct conditions or modes the robot can be in. Each state represents a specific action or set of actions.
  - **Transitions:** Rules or conditions that dictate when and how the robot moves from one state to another.
  - **Transition conditions:** External or internal signals (input/events) that trigger transitions between states.
  - **Initial state:** The starting state of the state machine.

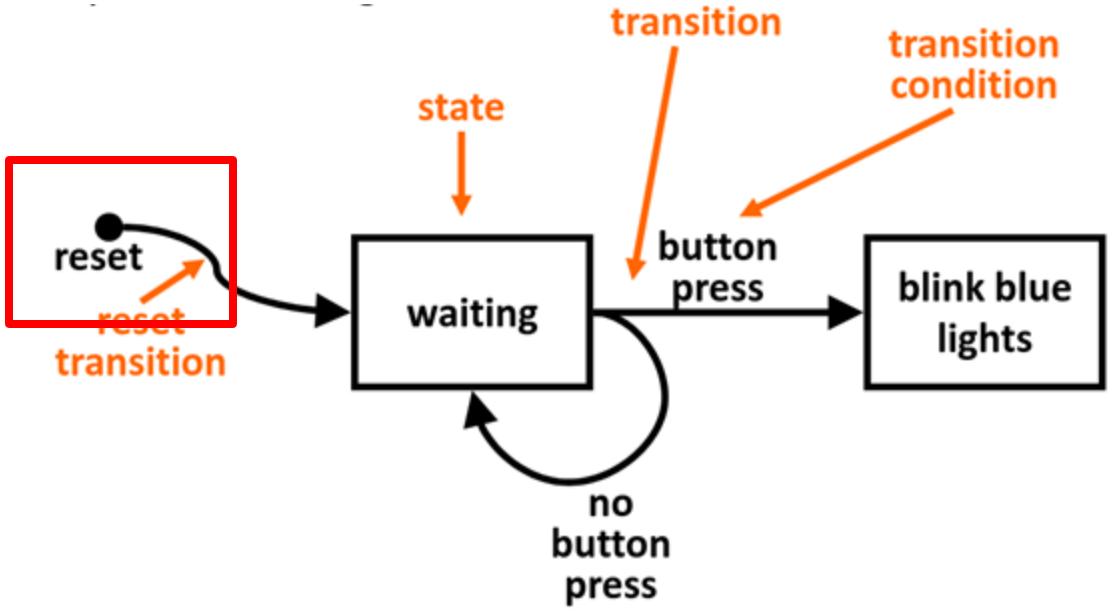
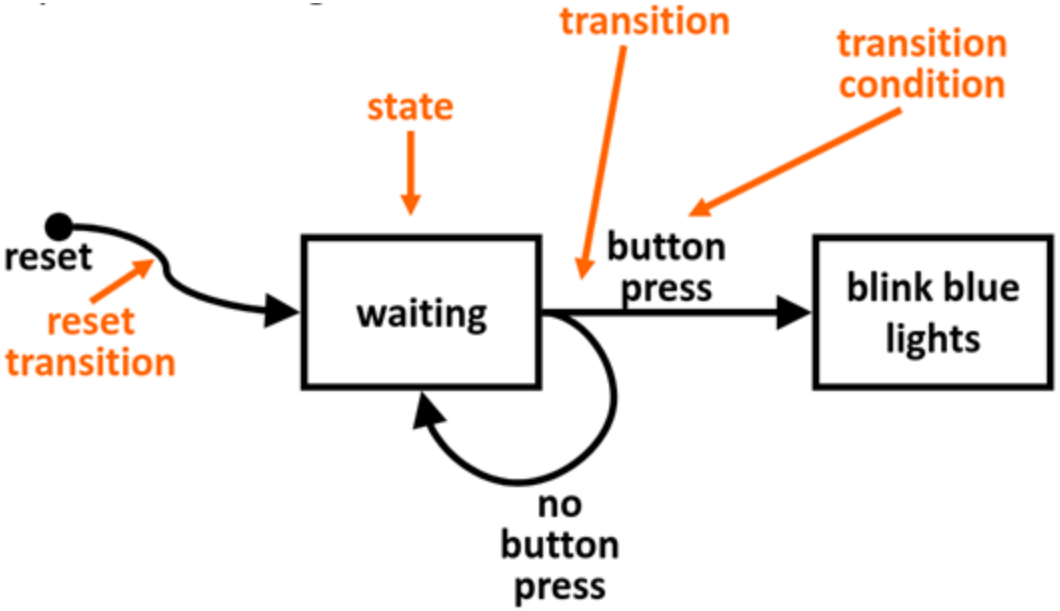


Illustration: https://www.clear.rice.edu/engi128/Handouts/Lec17-Robotics.pdf
https://en.wikipedia.org/wiki/Finite-state_machine

Finite-State Machines

*"A mathematical model of computation used to design systems that can be in one of a **finite** number of states at any given time."*



- **Components** (basic)
  - **States:** Distinct conditions or modes the robot can be in. Each state represents a specific action or set of actions.
  - **Transitions:** Rules or conditions that dictate when and how the robot moves from one state to another.
  - **Transition conditions:** External or internal signals (input/events) that trigger transitions between states.
  - **Initial state:** The starting state of the state machine.

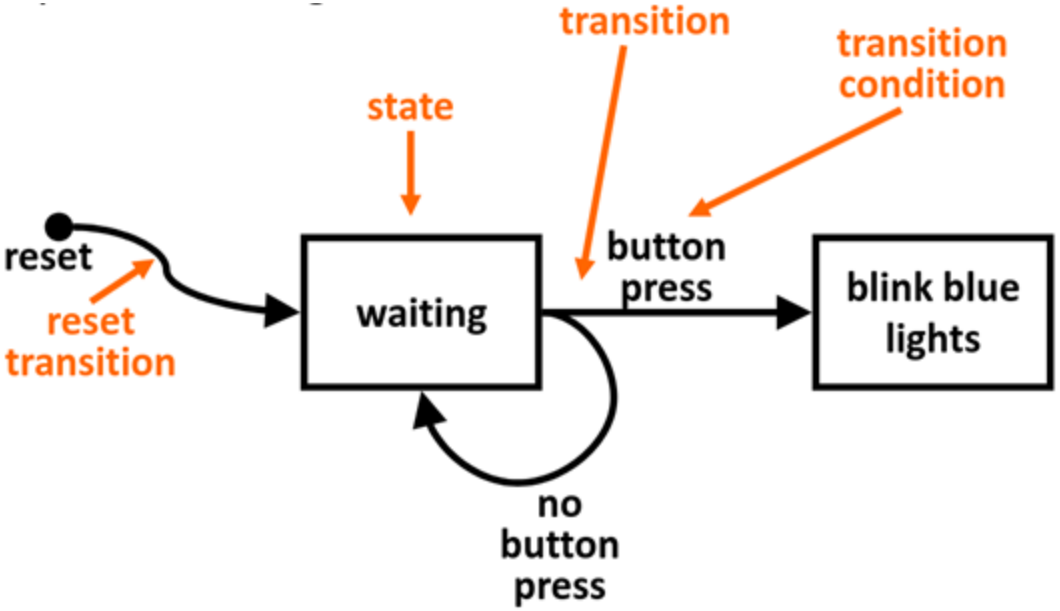
Example: A turnstile

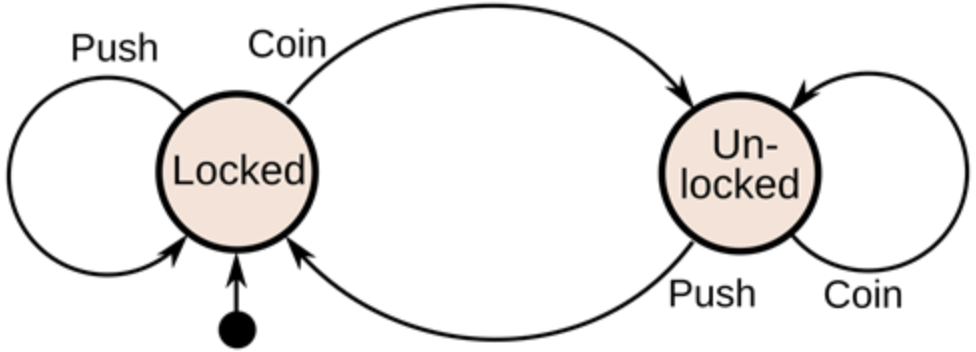
Illustration: https://www.clear.rice.edu/engi128/Handouts/Lec17-Robotics.pdf
https://en.wikipedia.org/wiki/Finite-state_machine

Finite-State Machines

*"A mathematical model of computation used to design systems that can be in one of a **finite** number of states at any given time."*

- **Components** (basic)
  - **States:** Distinct conditions or modes the robot can be in. Each state represents a specific action or set of actions.
  - **Transitions:** Rules or conditions that dictate when and how the robot moves from one state to another.
  - **Transition conditions:** External or internal signals (input/events) that trigger transitions between states.
  - **Initial state:** The starting state of the state machine.





Example: A turnstile

Illustration: https://www.clear.rice.edu/engi128/Handouts/Lec17-Robotics.pdf
https://en.wikipedia.org/wiki/Finite-state_machine

Finite-State Machines

*"A mathematical model of computation used to design systems that can be in one of a **finite** number of states at any given time."*

- **How to define a Finite-State Machine?**
  - ***Example:*** Wall detection

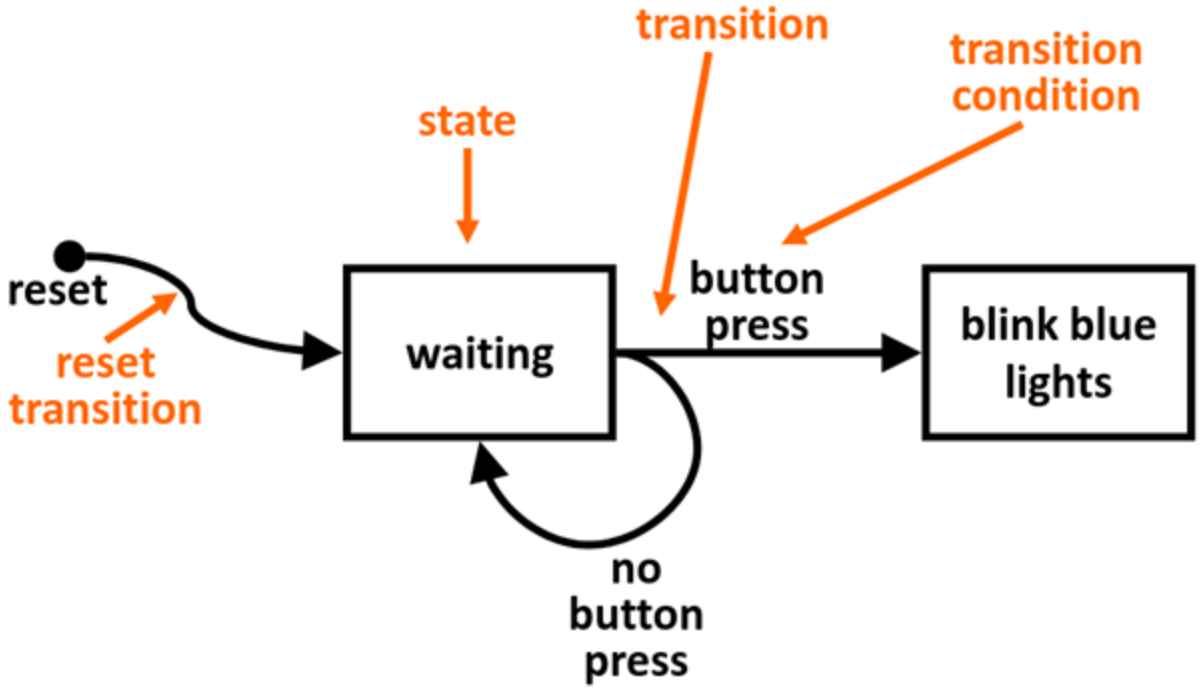


Illustration: https://www.clear.rice.edu/engi128/Handouts/Lec17-Robotics.pdf

Finite-State Machines (***Example:*** Wall detection)

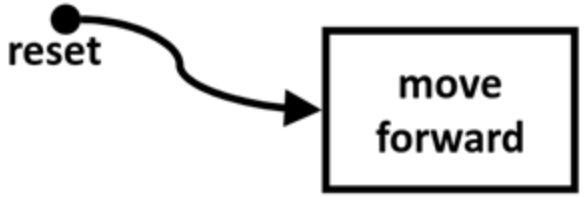


Illustration: https://www.clear.rice.edu/engi128/Handouts/Lec17-Robotics.pdf

Finite-State Machines (***Example:*** Wall detection)

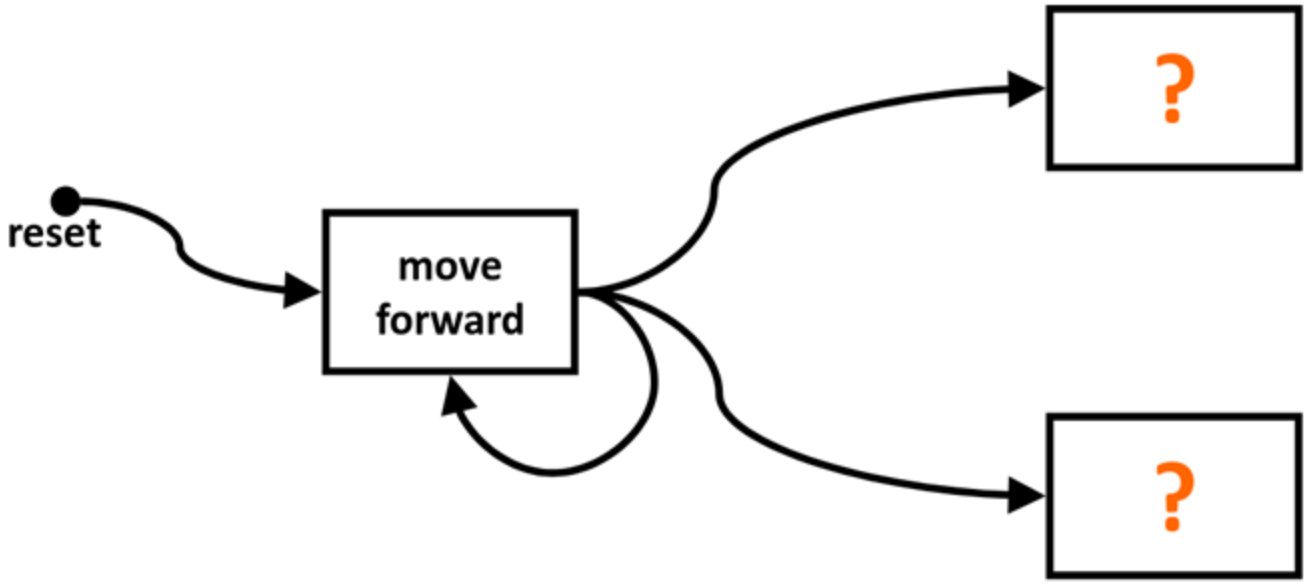


Illustration: https://www.clear.rice.edu/engi128/Handouts/Lec17-Robotics.pdf

Finite-State Machines (***Example:*** Wall detection)

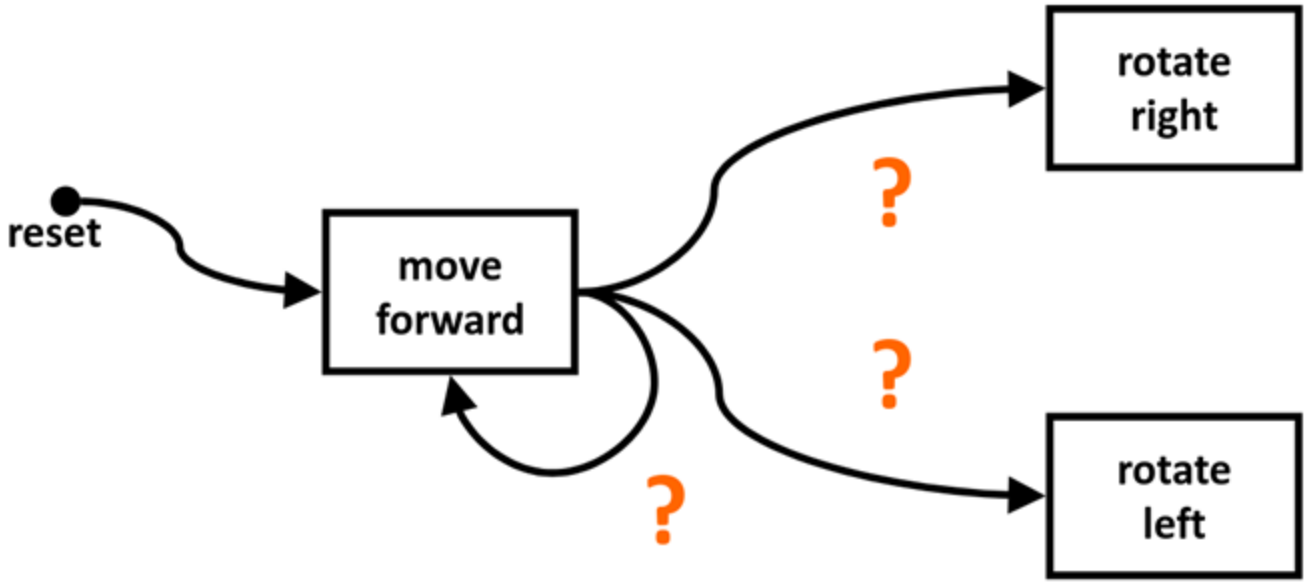


Illustration: https://www.clear.rice.edu/engi128/Handouts/Lec17-Robotics.pdf

Finite-State Machines (***Example:*** Wall detection)

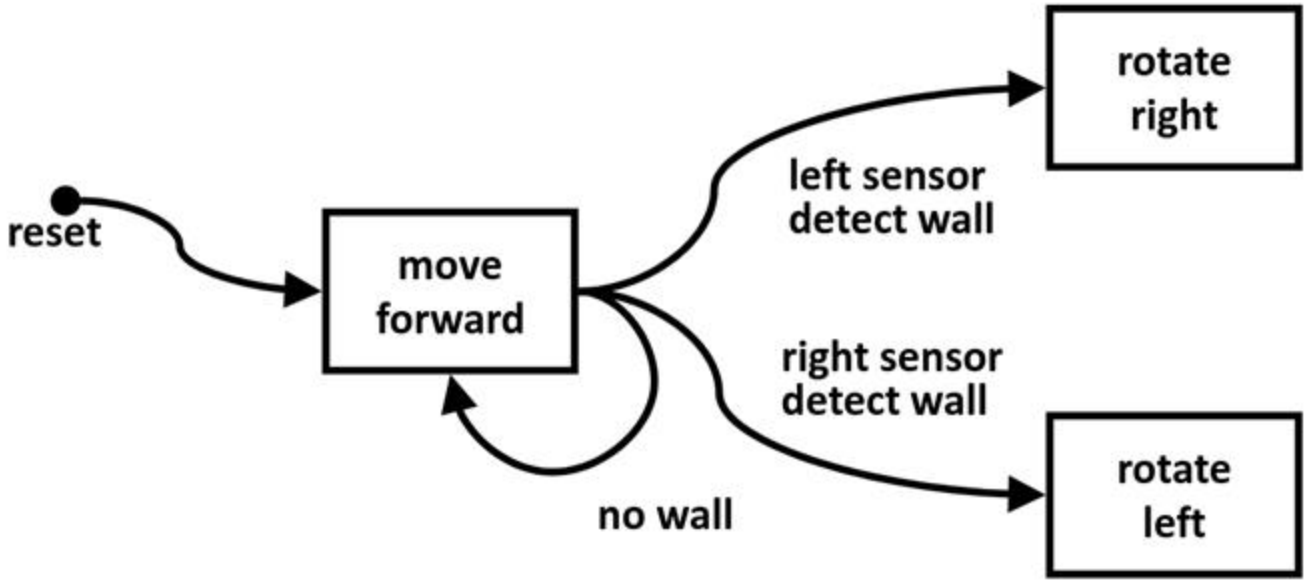


Illustration: https://www.clear.rice.edu/engi128/Handouts/Lec17-Robotics.pdf

Finite-State Machines (***Example:*** Wall detection)

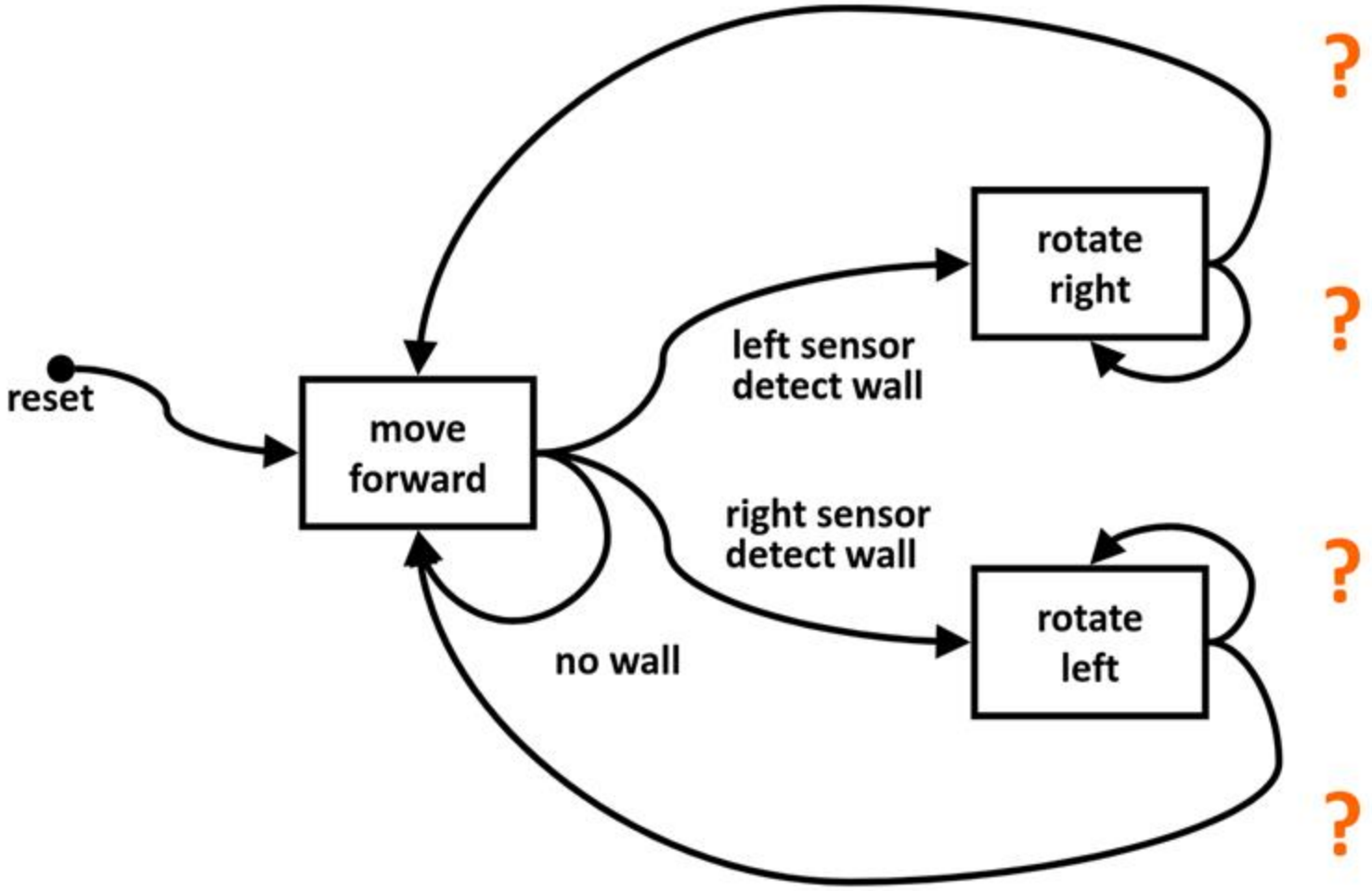


Illustration: https://www.clear.rice.edu/engi128/Handouts/Lec17-Robotics.pdf

Finite-State Machines (***Example:*** Wall detection)

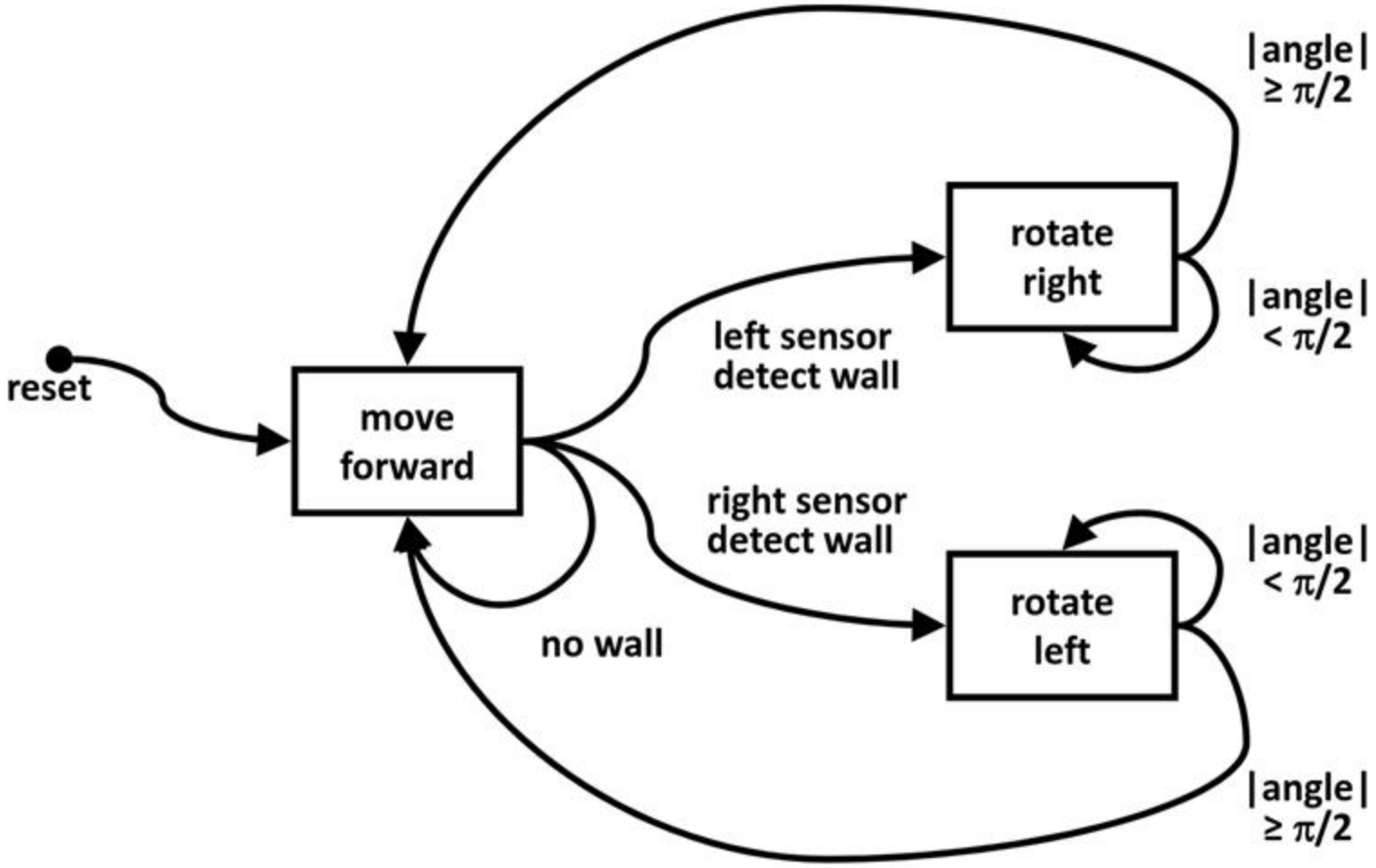


Illustration: https://www.clear.rice.edu/engi128/Handouts/Lec17-Robotics.pdf

Flowcharts

Flowcharts

“Visual diagrams that represent the flow of a process or system.”

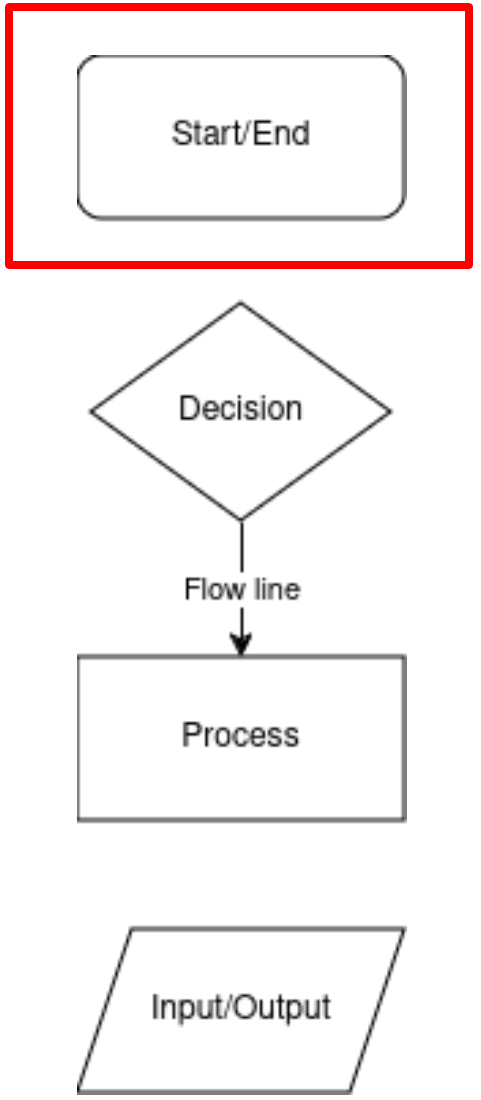
= Used to design and manage the behavior of mobile robots

Flowcharts

"Visual diagrams that represent the flow of a process or system."

= Used to design and manage the behavior of mobile robots

- **Components** (Basics)
  - **Start/End (Terminator):** Oval or rounded rectangle.

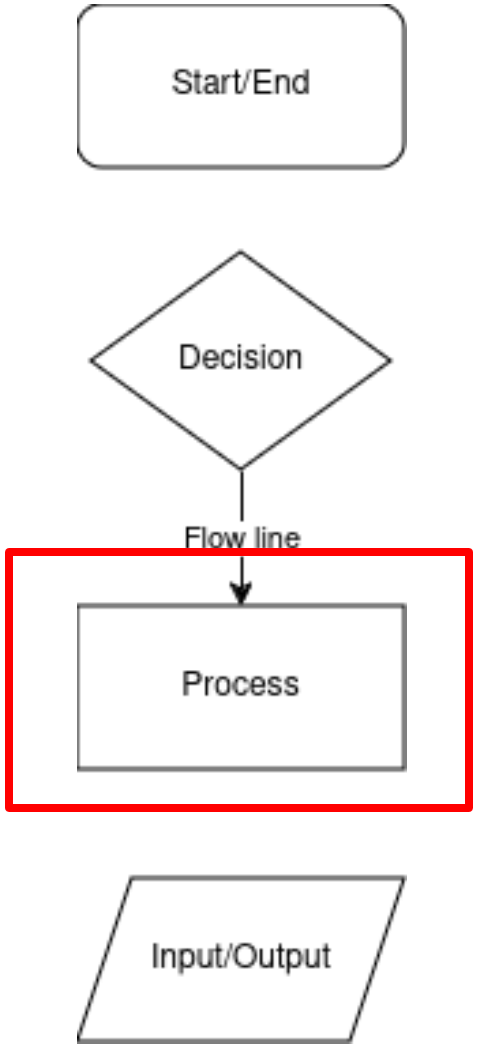


Flowcharts

"Visual diagrams that represent the flow of a process or system."

= Used to design and manage the behavior of mobile robots

- **Components** (Basics)
  - **Start/End (Terminator):** Oval or rounded rectangle.
  - **Process (Action/Operation):** Rectangle

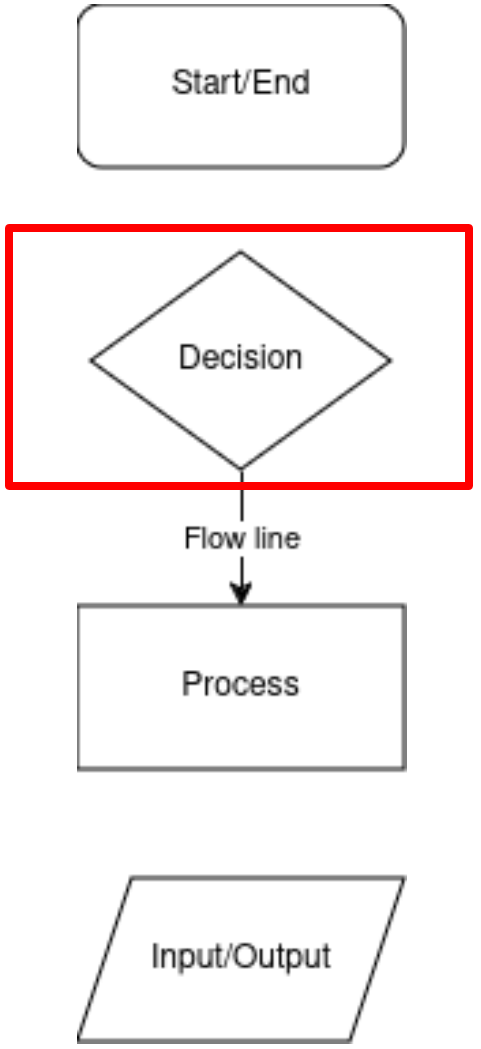


Flowcharts

"Visual diagrams that represent the flow of a process or system."

= Used to design and manage the behavior of mobile robots

- **Components** (Basics)
  - **Start/End (Terminator):** Oval or rounded rectangle.
  - **Process (Action/Operation):** Rectangle
  - **Decision:** Diamond

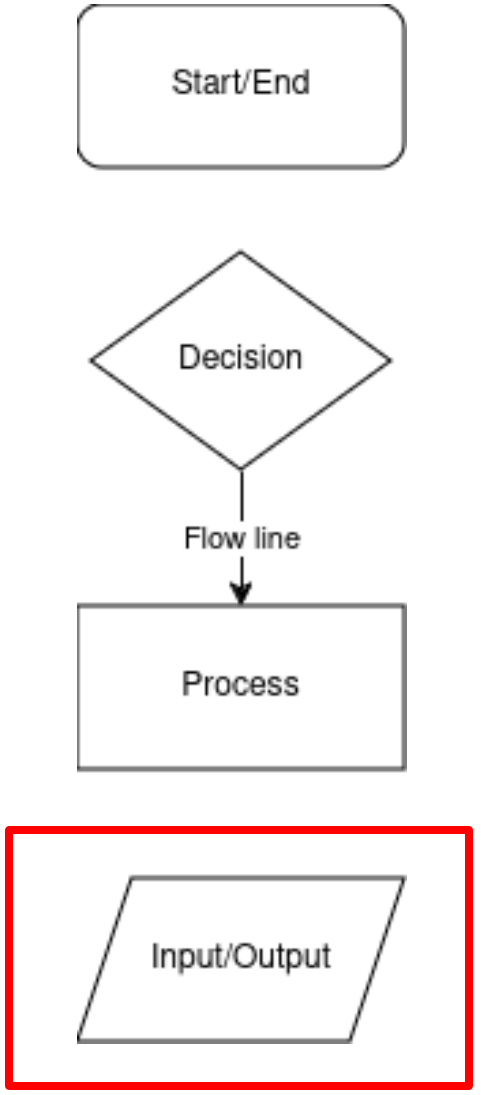


Flowcharts

"Visual diagrams that represent the flow of a process or system."

= Used to design and manage the behavior of mobile robots

- **Components** (Basics)
  - **Start/End (Terminator):** Oval or rounded rectangle.
  - **Process (Action/Operation):** Rectangle
  - **Decision:** Diamond
  - **Input/Output:** Parallelogram.

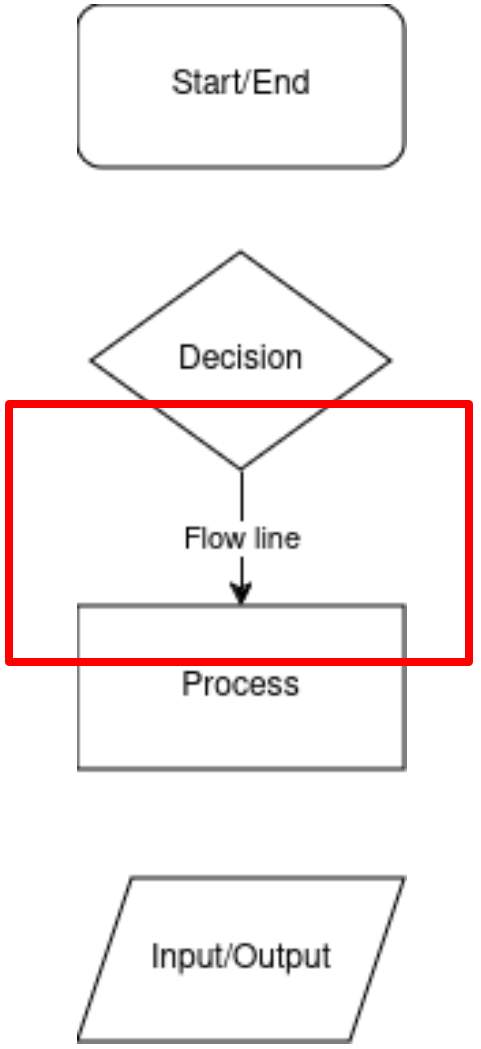


Flowcharts

"Visual diagrams that represent the flow of a process or system."

= Used to design and manage the behavior of mobile robots

- **Components** (Basics)
  - **Start/End (Terminator):** Oval or rounded rectangle.
  - **Process (Action/Operation):** Rectangle
  - **Decision:** Diamond
  - **Input/Output:** Parallelogram.
  - **Flow Line:** Arrow

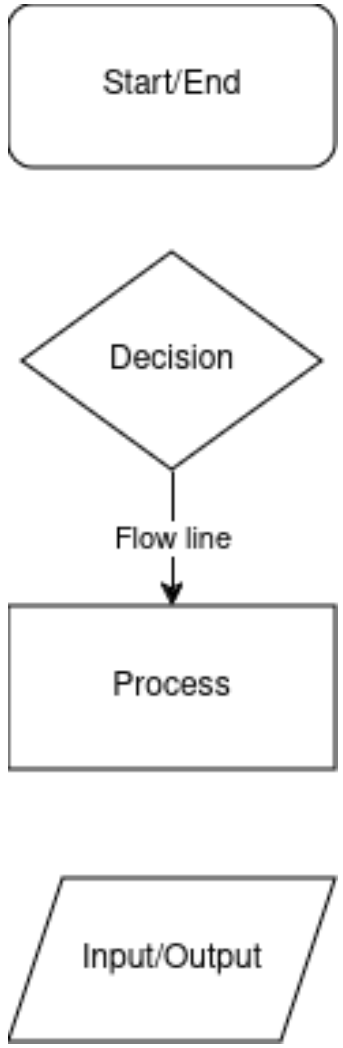


Flowcharts

"Visual diagrams that represent the flow of a process or system."

= Used to design and manage the behavior of mobile robots

- **Components** (Basics)
  - **Start/End (Terminator):** Oval or rounded rectangle.
  - **Process (Action/Operation):** Rectangle
  - **Decision:** Diamond
  - **Input/Output:** Parallelogram.
  - **Flow Line:** Arrow
  - …and many more…

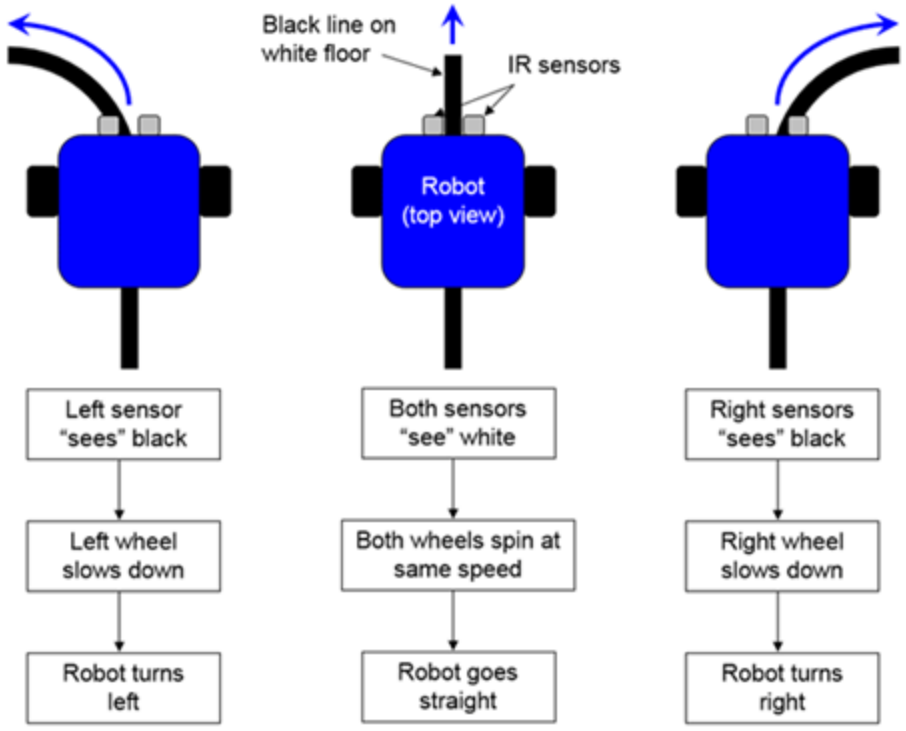


Tools (general use…)**:** Pen and Paper, DrawIO, TikZ, etc.

Flowcharts

"Visual diagrams that represent the flow of a process or system."

- **Example:** Move forward → Follow line?



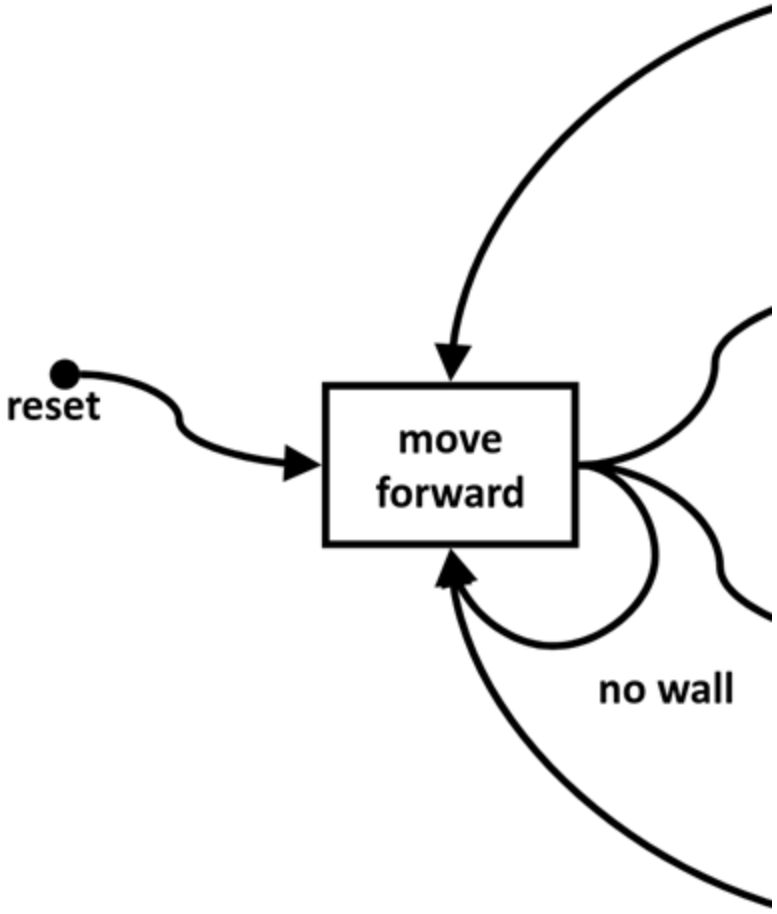
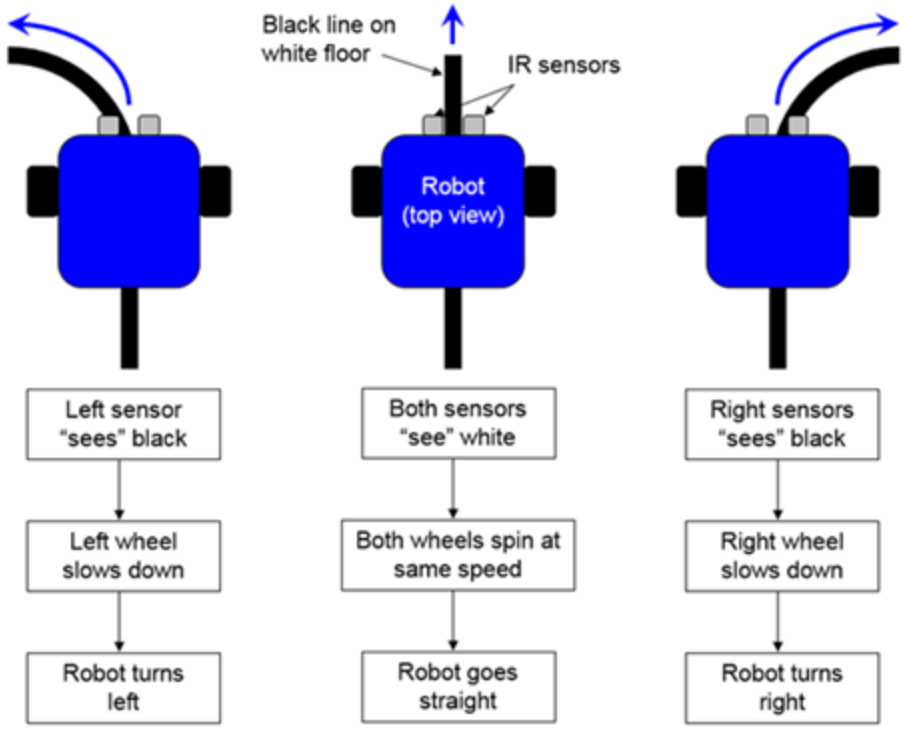


Illustration: https://www.sciencebuddies.org/science-fair-projects/project-ideas/Robotics_p023/robotics/line-following-robot

Flowcharts

"Visual diagrams that represent the flow of a process or system."

- **Example:** Follow line



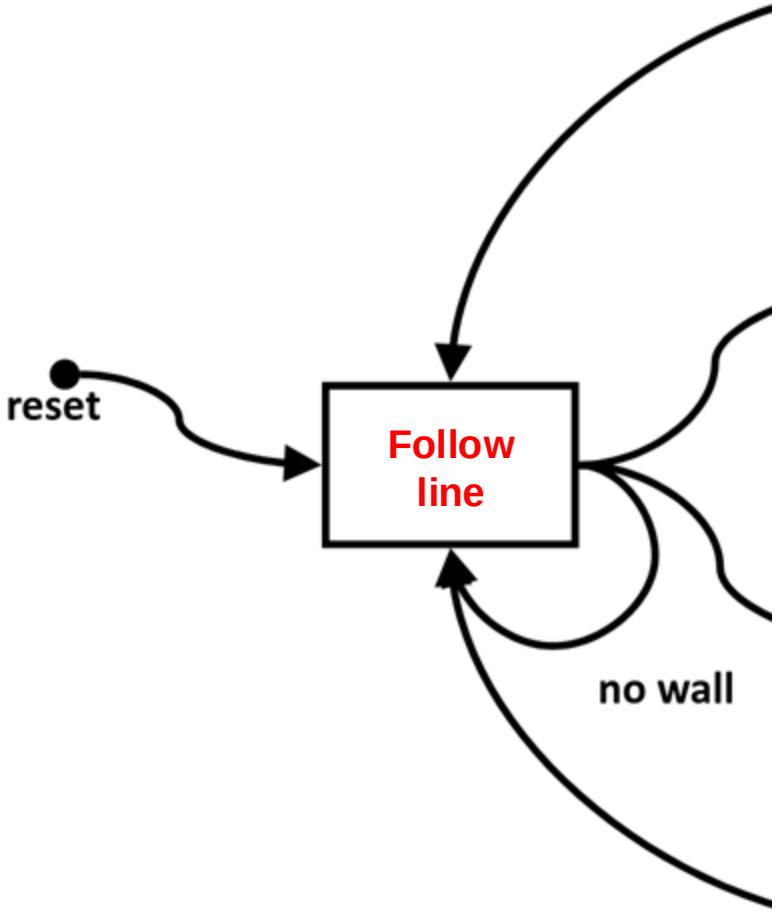
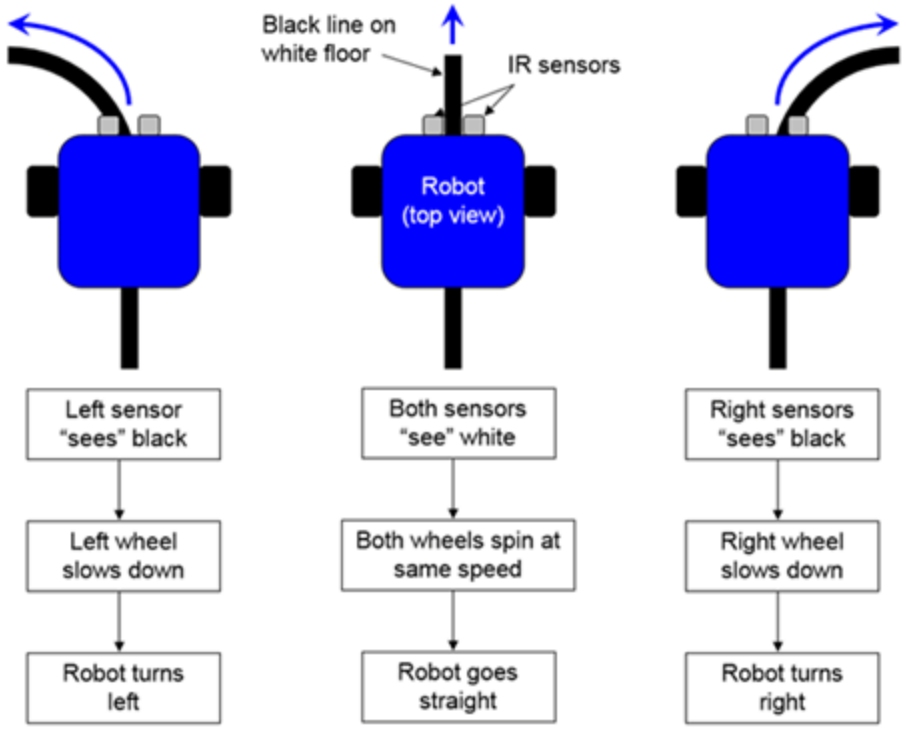


Illustration: https://www.sciencebuddies.org/science-fair-projects/project-ideas/Robotics_p023/robotics/line-following-robot

Flowcharts

"Visual diagrams that represent the flow of a process or system."

- **Example:** Follow line



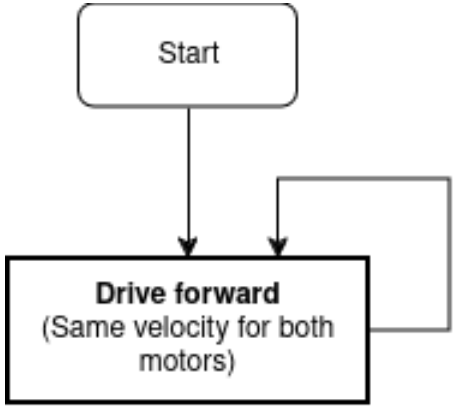
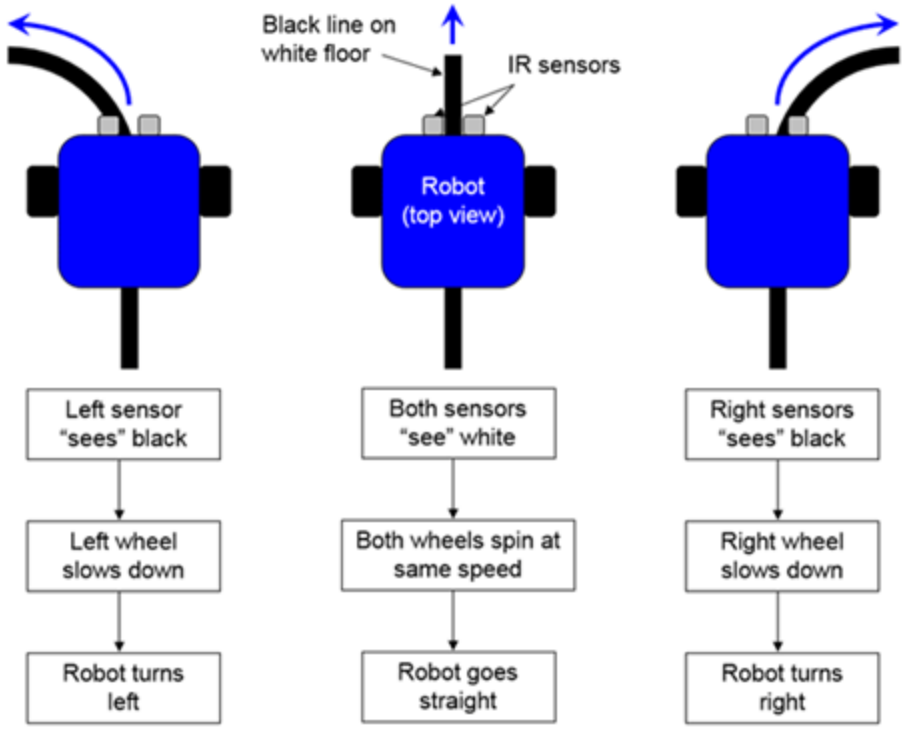


Illustration: https://www.sciencebuddies.org/science-fair-projects/project-ideas/Robotics_p023/robotics/line-following-robot

Flowcharts

"Visual diagrams that represent the flow of a process or system."

- **Example:** Follow line



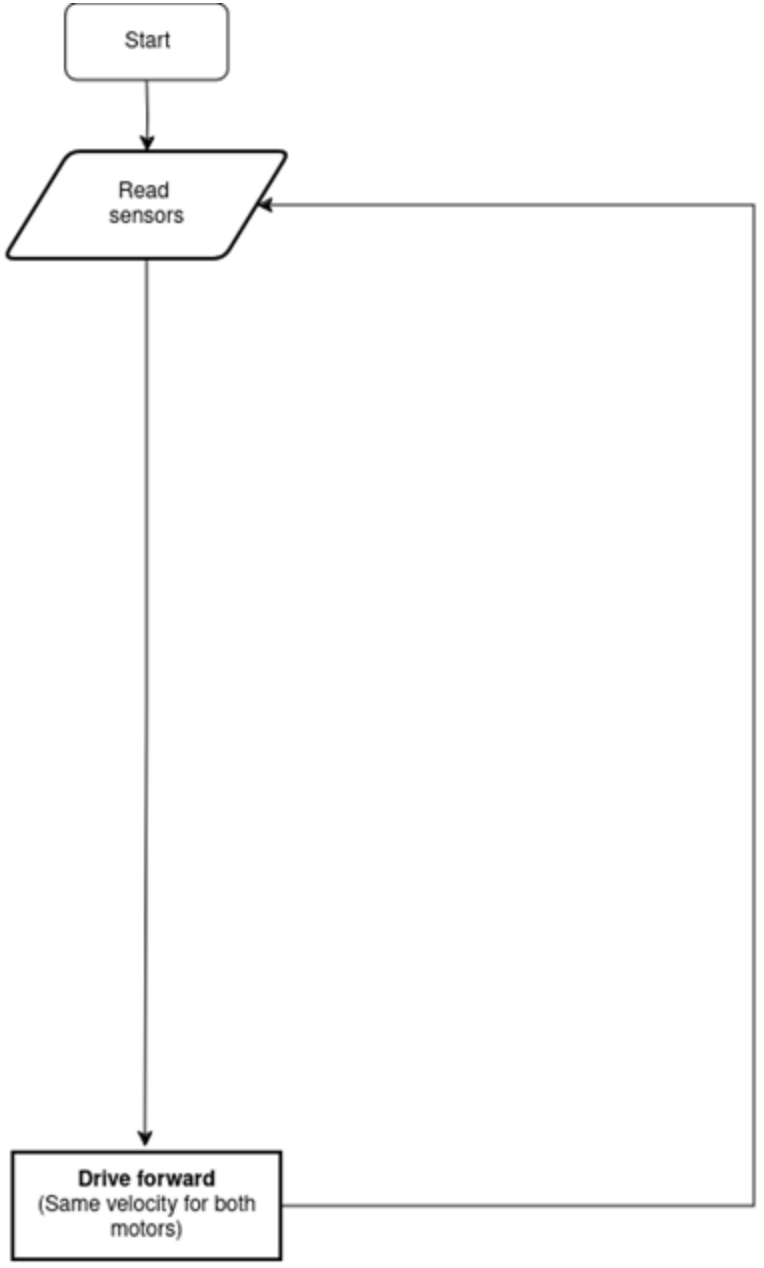
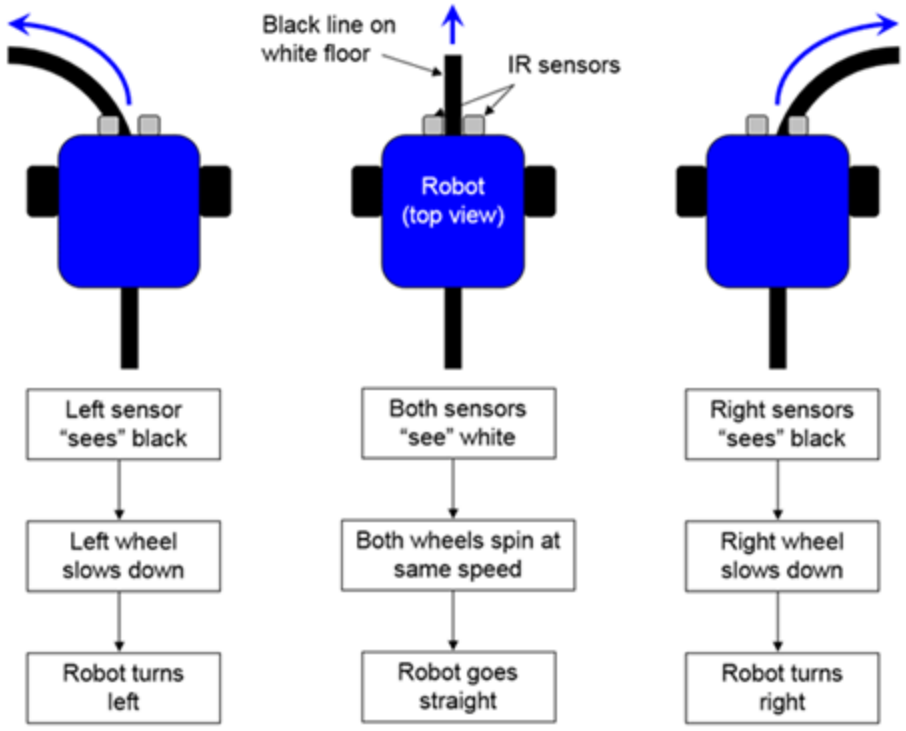


Illustration: https://www.sciencebuddies.org/science-fair-projects/project-ideas/Robotics_p023/robotics/line-following-robot

Flowcharts

"Visual diagrams that represent the flow of a process or system."

- **Example:** Follow line



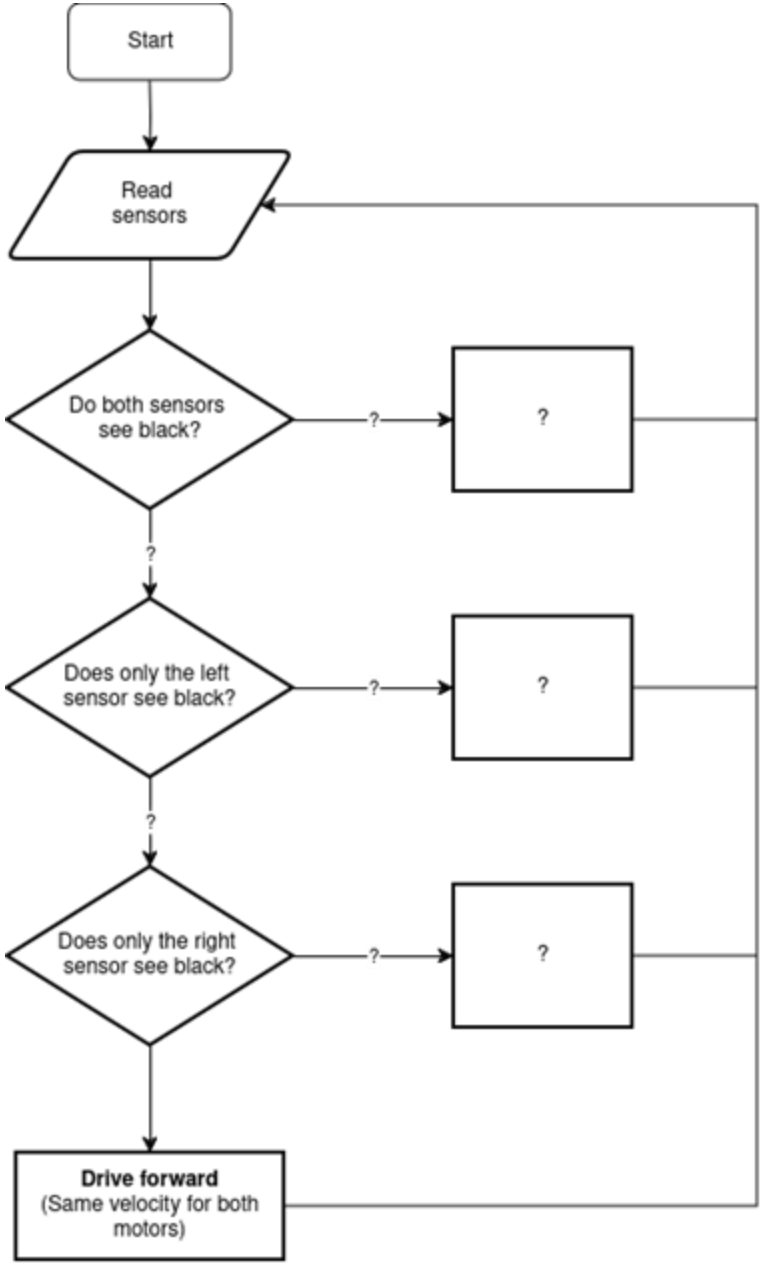
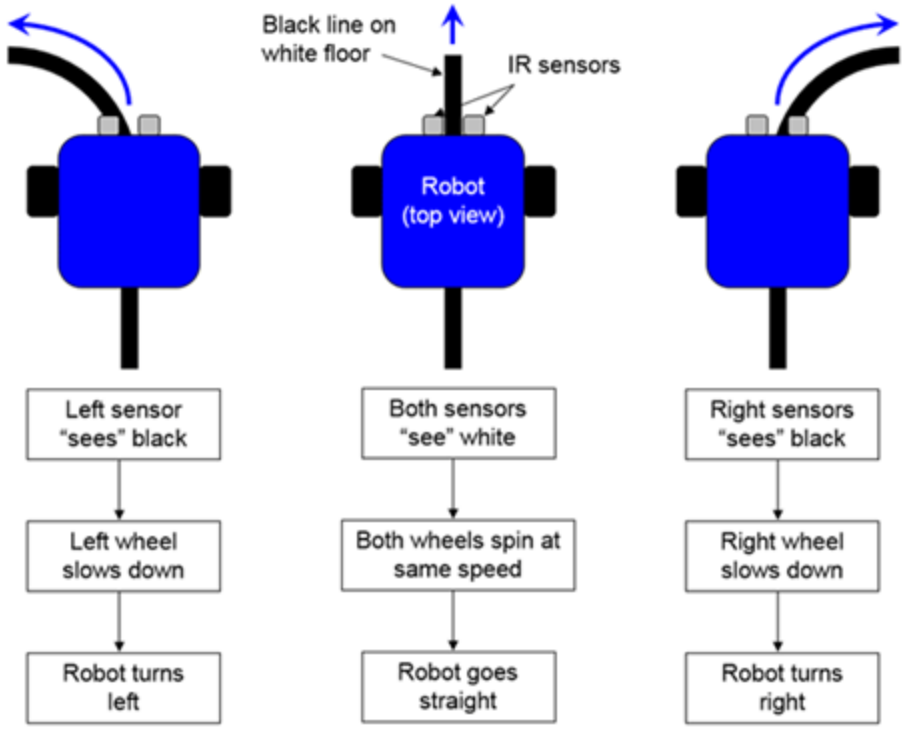


Illustration: https://www.sciencebuddies.org/science-fair-projects/project-ideas/Robotics_p023/robotics/line-following-robot

Flowcharts

"Visual diagrams that represent the flow of a process or system."

- **Example:** Follow line



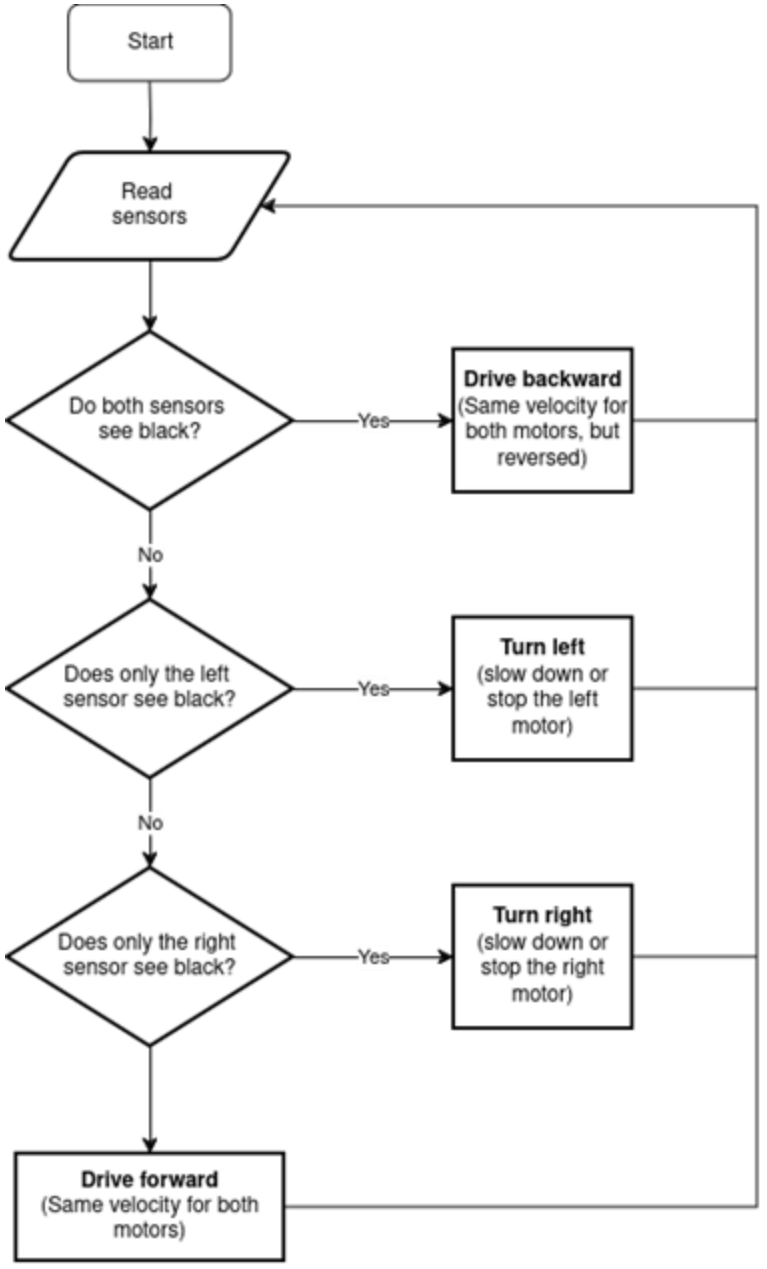
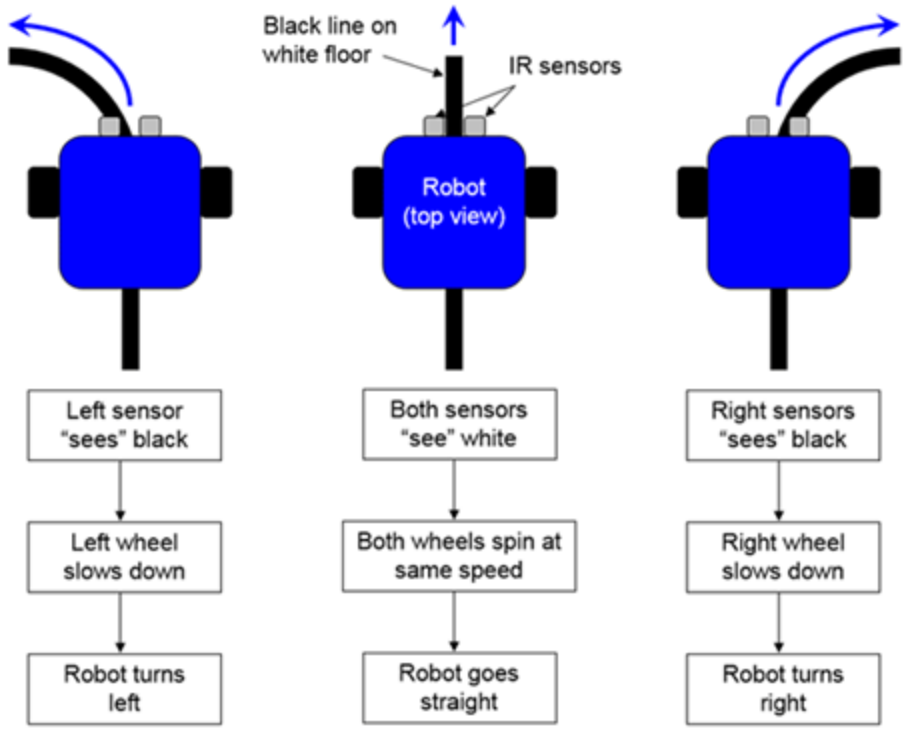


Illustration: https://www.sciencebuddies.org/science-fair-projects/project-ideas/Robotics_p023/robotics/line-following-robot

Feedback control

"...bringing together actuators and sensors through the software control algorithm"

= the actions depend directly on the feedback received from sensors like bumpers or distance/ToF sensors.



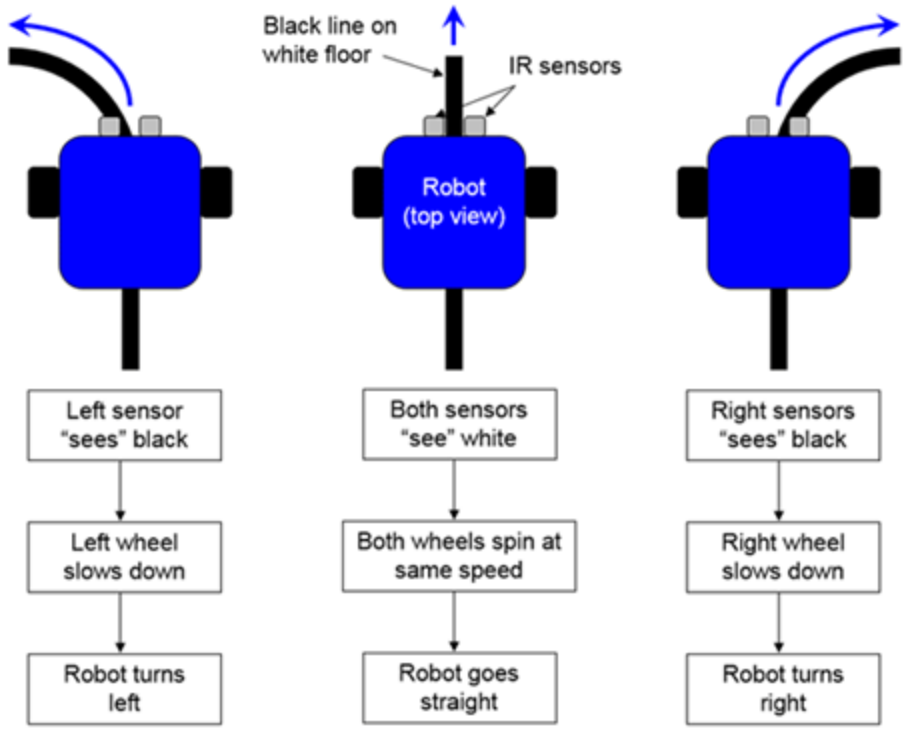
- Types
  - On-Off (or Bang-Bang) Control
  - (Beyond scope) Proportional Integral Derivative (PID) Control
  - (Beyond scope) Model Predictive Controller (MPC)
  - …and many, many more (**Control theory**)

Illustration: https://www.sciencebuddies.org/science-fair-projects/project-ideas/Robotics_p023/robotics/line-following-robot

Feedback control

"...bringing together actuators and sensors through the software control algorithm"

= the actions depend directly on the feedback received from sensors like bumpers or distance/ToF sensors.



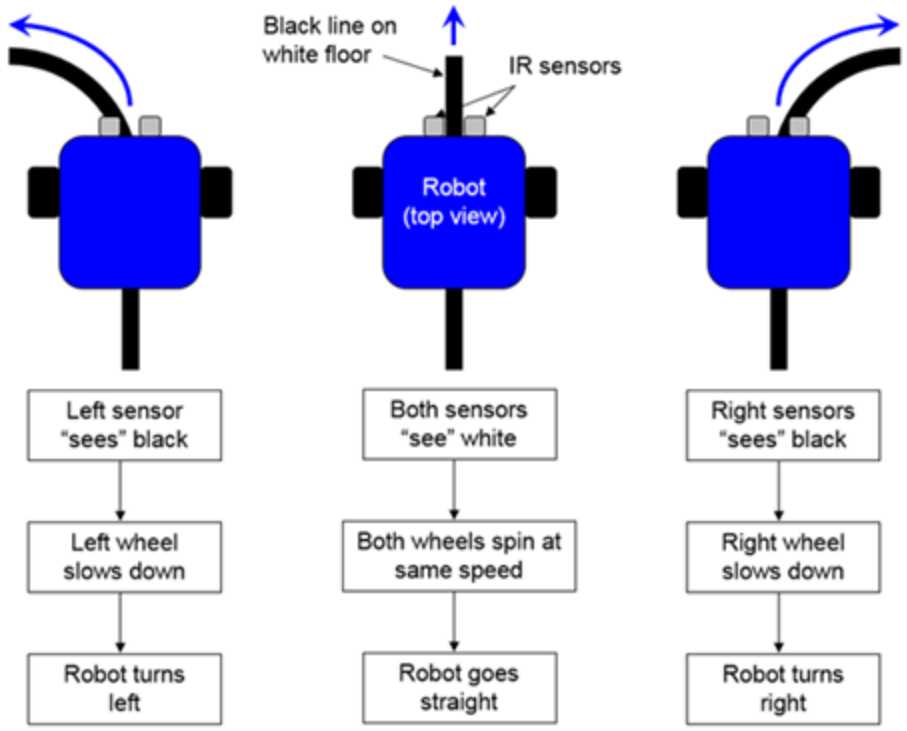
- Types
  - On-Off (or Bang-Bang) Control
  - ~~(Beyond scope) Proportional Integral Derivative (PID) Control~~
  - ~~(Beyond scope) Model Predictive Controller (MPC)~~
  - ~~…and many, many more (**Control theory**)~~

Illustration: https://www.sciencebuddies.org/science-fair-projects/project-ideas/Robotics_p023/robotics/line-following-robot

Feedback control

"...bringing together actuators and sensors through the software control algorithm"

= the actions depend directly on the feedback received from sensors like bumpers or distance/ToF sensors.



- Types
  - **On-Off (or Bang-Bang) Control**
    - The simplest control method
    - **Switches between two states**
      - eg. on and off for controlling a motor based on feedback, not PWM…
    - Operates on a threshold basis, eg.
      - If is is below the threshold, the control turns on.
      - If it exceeds the threshold, the control turns off.

Illustration: https://www.sciencebuddies.org/science-fair-projects/project-ideas/Robotics_p023/robotics/line-following-robot

Feedback control

"...bringing together actuators and sensors through the software control algorithm"

= the actions depend directly on the feedback received from sensors like bumpers or distance/ToF sensors.

- Types
  - **On-Off (or Bang-Bang) Control**
    - **Hysteresis**: …prevent frequent switching (or oscillations) between the "on" and "off" states…
      - Uses two different (dual) thresholds;
        - **Upper Threshold:** The controller turns off when the system variable reaches or exceeds this value.
        - **Lower Threshold:** The controller turns on when the system variable falls below this value.
        - **Hysteresis band:** The gap between the two

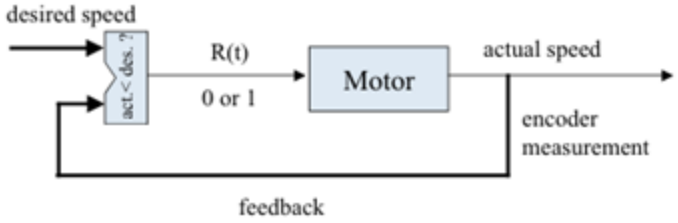


Figure 7.1: On-off controller

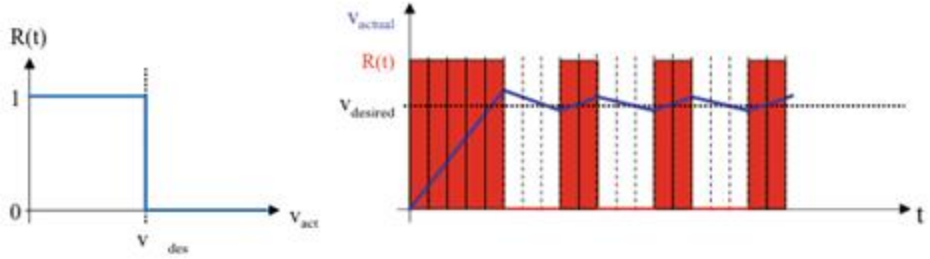


Figure 7.2: On-off control signal: control signal (left); motor output over time (right)

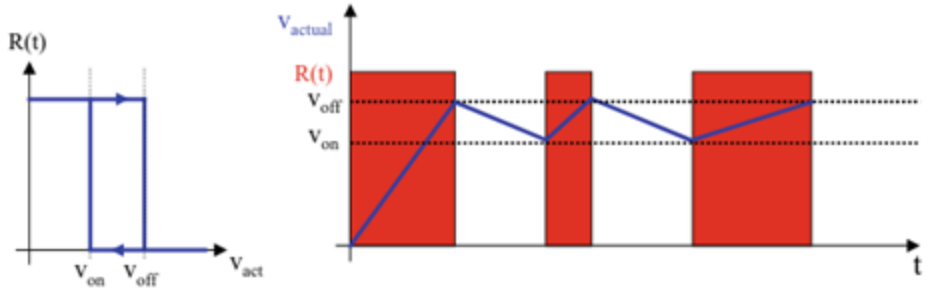
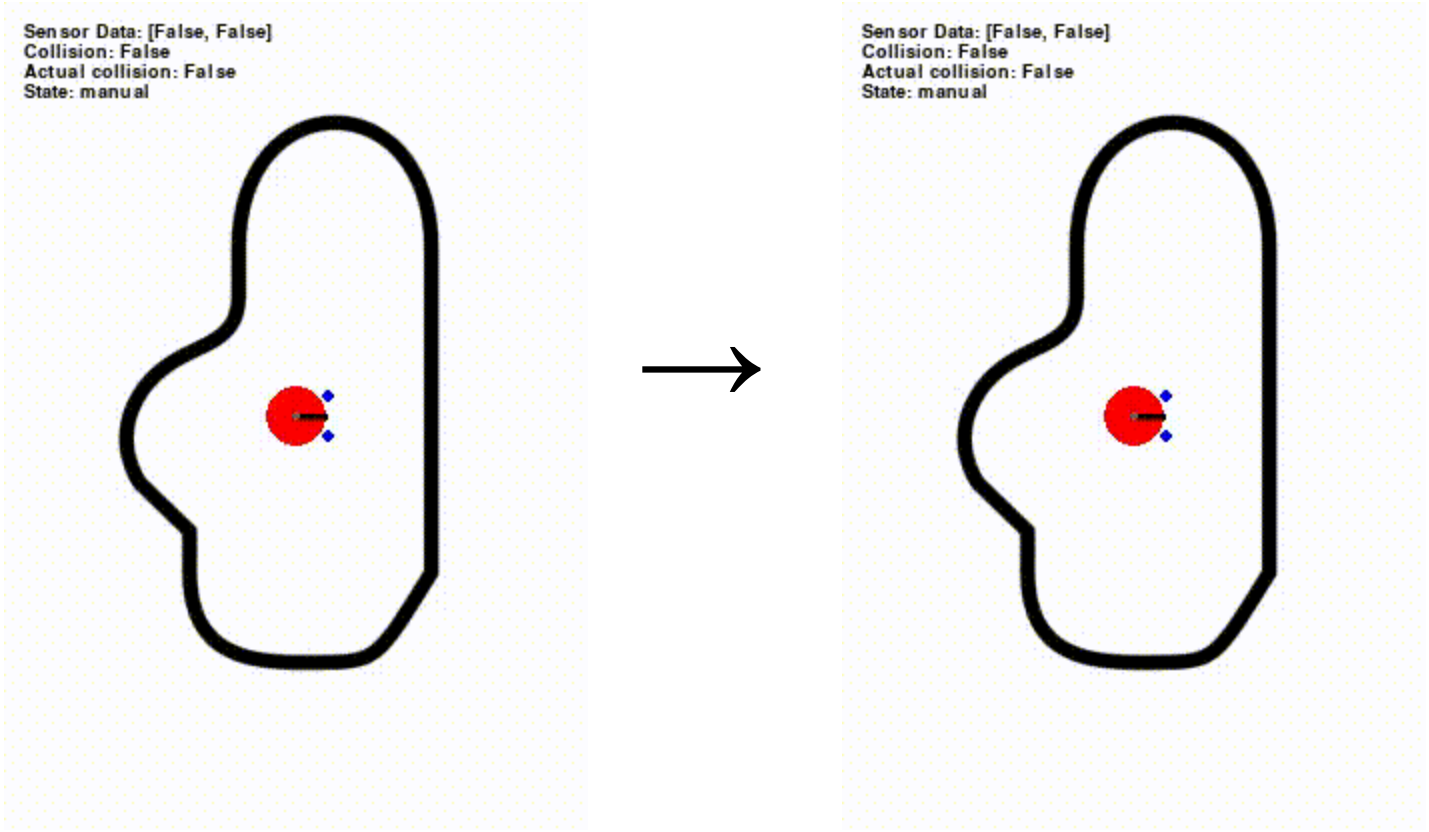


Figure 7.3: On-off control signal with hysteresis band (left); motor output over time (right)

Illustrations: [eB1] From Mobile Robots to Autonomous Vehicles with Raspberry Pi and Arduino

Extra Credit Activities #3:
Random Walk behaviour



Assignments