(SER)^2 Workshop on Robotics

Introduction

Robotics requires skills in many fields: electronics, computer science, mechanical engineering, and control theory. Open-source software and hardware communities provide students of robotics many opportunities to get started. Without these, the many years of study required to become conversant in all these fields would be discouraging. Inexpensive, off-the-shelf units can perform highly complex functions without requiring the student to fully understand the details, giving access to exciting applications to the novice.

The student can begin to design immediately after learning a few basics. In this workshop, we will outline those basics and delve a bit deeper into a few. Armed with this understanding and open-source software and hardware, a student can begin to design and prototype robotic systems.

Objectives

It is the primary objective of this workshop to deepen middle and high school STEM teachers' understanding of robotics. The secondary objective is to provide the teachers resources for further exploration.

Defining robotics

The meaning of the word "robot" has evolved since its popular introduction in a Czech play in 1921.[1] Even preceding the coinage, people had long built machines that "worked" for humans. However, these were not robots in the modern sense.

Here is a definition of robotics:[2]

> *A robot is an autonomous system which exists in the physical world, can sense its environment, and can act on it to achieve some goals.*

I prefer to interpolate the word **synthetic** in the definition, since, without it, this definition could describe an animal or even a human.

A robot is **autonomous** in that, given sensory inputs, it chooses without any explicit user input how to act. This eliminates many machines from the definition. However, the line is somewhat blurry, and there seems to be more of a spectrum of autonomy. Consider the scissors. A simple mechanism, to be sure. But what separates it from a simple control system like a thermostat? The thermostat has sensor inputs and it can turn on and off a furnace. But a human had to set the thermostat, and the basic if-then logic of the thermostat is hardly free-thinking. However, surely this human input is farther-removed from the fully manual scissors. Machines, then, are might be classified as more autonomous as their requirement of human input in order to function properly recedes.

Although we do not wish to introduce a physical/other dichotomy, we would like to require that a robot must exist in the **physical world**. This requirement is there primarily to eliminate from consideration *simulations* of robots. Computer programs can be used to simulate a robot, but these are not robots, themselves. Of course, the pedantic have a valid point here: does not a simulation exist in the physical world? Well, yes, of course it does. We probably mean something closer to "can act on substances outside itself," which would encompass systems that influence the state of an electrical circuit, for instance. We must take care not to limit robotics in our definition, but we must also strive for clarity and flee from nonsense.

A robot can **sense** its environment. We typically separate user *inputs* from environmental *sensations*. This means that user's button is not a sensor. **Sensors** give the robot some sort of information about its environment. There are lots of sensors available, including visual, auditory, touch, pressure, speed, distance, strain, light, and temperature sensors.

A robot can **act** on its environment. This requirement means the robot must have some sort of **actuator**.

Now, if our robot just acted on its environment at random, it would totally suck and we'd probably tell it it's no real robot at all. Therefore, we say that a robot must act on its environment in order to **achieve some goals**. This is why **artificial intelligence** is such a big deal. It allows the robot to be increasingly autonomous and yet exhibit rational behavior.

Discussion questions

1. Why might we want to differentiate between simulated and real robots so sharply?
2. What is intelligence?
3. How might an animal display intelligence?
4. How might a robot display intelligence?

History of robotics

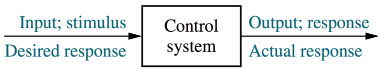
The practical possibility of robotics is based on developments in three fields: control theory, cybernetics, and artificial intelligence.

Control theory

A standard definition of **control theory** is:[3]

> *A control system consists of subsystems and processes (or plants) assembled for the purpose of obtaining a desired output with desired performance, given a specified input.*

The following block diagram is helpful when considering this definition.[3]



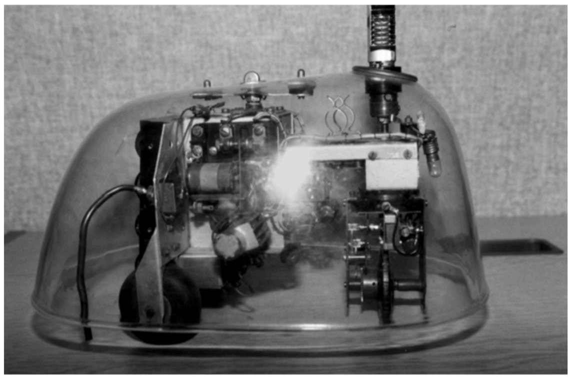
Without control theory, there would be no robotics. Control systems are found in many biological systems that vastly predate humans. However, human-made control systems date back to around 300 BC—the Greeks built a water clock that included a float valve, similar to those in modern toilets, to control the volume of water in a tank. Liquid-level control was the only type of control system until the 17th century brought steam pressure and temperature controllers. In the 18th century speed controllers were developed for windmills and steam engines.

Control theory as we understand it today began in the second half of the 19th century with Maxwell, Routh, and Lyapunov. They began applying mathematics extensively to control systems, and some of their methods are still in use today. In the 20th century, the gyroscope was used for automatic ship steering; Minorsky developed what would become PID control; Bode and Nyquist developed the analysis of feedback amplifiers and frequency response controller design techniques; and Evans developed root locus techniques in 1948. The theories and techniques from this period are typically called **classical control theory**. Finally in the late 1950s and early 1960s, what is known as **modern control theory** began, which is based on state-space methods. It is important to note that both classical and modern control theory are still widely used and complement each other well.[3]

Cybernetics

Cybernetics is "the scientific study of control and communication in the animal and the machine." [4] The general idea was to understand biological systems and create artificial systems that could perform autonomously in their environments, much like an animal or human. This field produced what is often considered to be the first robot: **Grey Walter's Tortoise**.

W. Grey Walter was a neurophysiologist in and after the 1940s. He built *biomimetic* turtle-like robots as shown below.



One of Walter's robots was named **Machina Speculatrix** ("machine that thinks"). It consisted of a photocell, a bump sensor, a rechargable battery, three motors, three wheels, and one analog circuit. It had the following capabilities:[2]

1. find the light,
2. head toward the light,

3. back away from bright light,
4. turn and push to avoid obstacles, and
5. recharge battery.

Walter's robots used **reactive control**—control that uses prioritized "reflexes" to determine behavior. Later, we will learn more about reactive control, which can create "animal-like" robotic behavior.

Some of the behavior of these robots were rather complex yet not programmed-in. This is now called **emergent behavior**.[5]

Cybernetics was later split into two fields: AI and robotics. AI focused on the "thinking" aspect and robotics on interaction with the environment. Mataric and others believes this was to the impediment of both fields, and it took some time for integrated thinking to return.[2]

Valentino Braitenberg was inspired by Walter's work to write the book *Vehicles* in 1984. *Vehicles* describes a number of *thought experiments* about the design of simple robots. He never built them, himself, but others have used his work as the basis of their own.

Artificial Intelligence

Artificial intelligence became an official field of study when Marvin Minsky, John McCarthy, Allan Newell, and Herbert Simon participated in a conference at Dartmouth University in 1956.[2]

> *The conclusions of the meeting can be summarized as follows: in order for machines to be intelligent, … they will need to use the following:*
>
> - *Internal models of the world*
> - *Search through possible solutions*
> - *Planning and reasoning to solve problems*
> - *Symbolic representation of information*
> - *Hierarchical system organization*
> - *Sequential program execution.*

AI-inspired robots built in the 70s and 80s were primarily focused on navigation and used a now-defunct type of control called **purely deliberative control**. Later, we will learn more about this and the more modern types of robot control: **reactive control**, **hybrid control**, and **behavior-based control**.

Of the three key fields of study that comprise much of the history of robotics—control theory, cybernetics, and AI—all three are still active areas of research, although the term "cybernetics" has gone out-of-style in favor of such terms as "biomimetic robots" and the like.[6]

Discussion questions

1. The symbol-manipulation based AI, sometimes called "good old-fashioned AI" (GOFAI), is still being developed, but mostly outside the field of robotics. Why might it have failed to achieve artificial intelligence in robots?
2. Emergent behavior is totally awesome but is usually discovered by accident. How might a designer design for emergence?

Robot components

Our definition of a robot suggests several types of components that are required for a robot instantiation:[2]

1. a **body** (so it has the potential to act on its environment),
2. **sensors** (so it can sense its environment),
3. **effectors** and **actuators** (so it can act on its environment), and
4. a **controller** (so it can be autonomous).

Let us consider each broad class of components, in turn.

Embodiment

Without a body, a robot could not act on its environment. Of course, a rock has a body, so embodiment alone does not make something a robot. The primary implication of embodiment for robots is that they are subject to the physical laws of nature. These are limitations, but also opportunities. They require power, can collide with things, and can take a long time to perform a task, *but* they can potentially harvest energy, avoid collisions, and perform tasks quickly. They are embroiled in the same buzzing, frothy mess of a world we are, and this makes designing a robot a serious challenge.

Sensing

Sensors are dynamic systems that interact with their environment and can indicate part of the state of their environment. Sensors allow robots to sense/perceive their environment. Depending on the robot's task, it will require certain types of sensors and not others. Later, we will consider several classes of sensors.

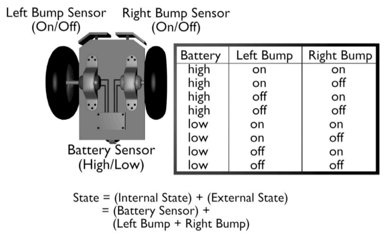
Using sensors, a robot is able to (at least partially) estimate its **internal state** in the system dynamical sense of a minimal set of variables that completely describe the robots's response at a given time. A robot can also be made aware of the state of its environment or **external state** through sensing. A robot's (and its environment's) state may or may not be completely described by its sensors. This gives three possibilities:

1. **observable**: state is completely sensed,
2. **partially observable**: state is partially sensed, or
3. **unobservable**: state is not sensed at all.

Typically, a robot is going to perform best when it has full observability and worst when it has full unobservability, but there are trade-offs to consider, as we will see when we further explore observability.

State variables, variables that describe state, can be either **discrete** (e.g. a binary field) or **continuous** (e.g. $\mathbf{x} \in \mathbb{R}^n$) variables.

State space is the set of all possible states. The figure below shows a table that includes the entire state space for a simple robot.[2]


Left Bump Sensor (On/Off)
Right Bump Sensor (On/Off)
Battery Sensor (High/Low)

Battery	Left Bump	Right Bump
high	on	on
high	on	off
high	off	on
high	off	off
low	on	on
low	on	off
low	off	on
low	off	off

State = (Internal State) + (External State)
= (Battery Sensor) +
(Left Bump + Right Bump)

A robot might have an **internal model** or **representation** that can be used to describe the environment.

A robot's **sensor space** (or **perceptual space**) is the set of all possible states it can sense.

Action

A robot's **actuators** move its **effectors** in order to interact with it's environment. The two primary interactions are **locomotion** (moving) and **manipulation** (handling things). In fact, these are the two primary subfields of robotics: **mobile robotics** (studies robots that move on the ground, and through the air and water) and **manipulator robotics** (studies robots that manipulate objects, typically via a robotic arm).

Power and its relation to computation

As with many other technologies—and especially mobile technologies—robots have important power demands. In animals, brains demand a relatively large amount of power relative to the rest of its body. In robots, processing power is relatively small and actuation power is relatively high.

Autonomy

Through hardware and software, **controllers** allow a robot to be autonomous. From sensor inputs, controllers determine actuator and therefore effector outputs. **Autonomy** is the ability to act in an environment without significant guidance from an outside user.

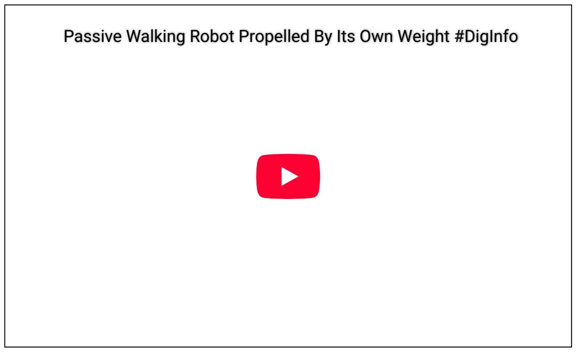
Effectors and Actuators

In order for a robot to act on its environment, it must have two key components: **effectors** and **actuators**. Actuators are the components that *drive* the effectors, which in turn *interact with* the environment. The signal flows from controller to actuator to effector to the environment; in fact, often aspects of the environment are detected by sensors, which send signals to a controller, which sends a signal to actuators, which actuate effectors, which affect the environment.

Examples of robotic actuators include electric motors, pneumatic cylinders, and piezoelectric actuators. Examples of effectors are arms, grippers, wheels, propellers, and fins.

Active vs passive actuation

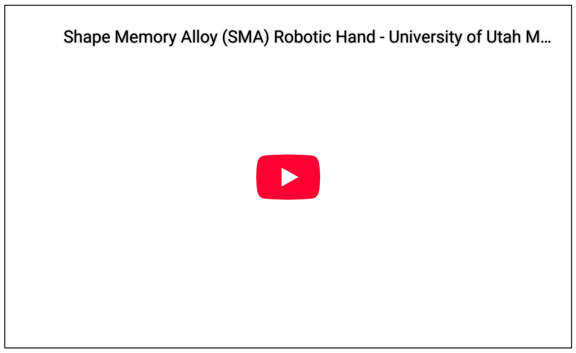
We call actuation that requires externally supplied power **active actuation** and actuation that requires only "free" potential energy from its environment **passive actuation**. The former is by far the most common, but there are some examples of passive actuation, including gliders and the totally awesome *Passive Walker* robot developed by Tad McGeer, shown in a recent instantiation, below.


Passive Walking Robot Propelled By Its Own Weight #DigInfo

Types of actuators

There are several types of actuators. Here is a description of some of the most common types. They all happen to be active.

- **Electric motors** are the most common robotic actuator. They can be relatively inexpensive and easy to use. We will consider them further, momentarily.
- **Hydraulic actuators** use fluid pressure to (typically) move a piston-like mechanism. These can be powerful (and dangerous) and are great for large load applications. They can develop leaks.
- **Pneumatic actuators** are similar to hydraulic actuators except that instead of fluid, they use pressurized gas (typically air). These can be very precise and work at extreme temperatures, but can handle much less load than hydraulic actuators. These also can develop leaks.
- **Smart material actuators** are actuators that usually create linear motion when activated by various mechanisms. Examples include chemically reactive, thermally reactive, and piezoelectric materials. Below is a video of a robotic hand that uses shape memory alloys.


Shape Memory Alloy (SMA) Robotic Hand - University of Utah M...

Motors

These are everywhere in robots. They all use electrical energy to turn a shaft (or actuate a rod if it is a linear motor), and are most ubiquitously used to rotate wheels and arms. Let's consider the most common types.

└ DC motors

Direct-current (dc) motors are the simplest, inexpensive, and easy to use type of motor, and they come in a large variety of sizes. They use dc electrical power, which is converted to rotational mechanical power via a **rotor** (or **armature**) and magnetic **stator**. There are two main types: brushed and brushless. The video below shows how a brushed dc motor works.

DC motors approximately obey the power conservation law $P_{elec} = P_{mech}$. Instantaneous electrical power is the product of the voltage v and current i, or $P_{elec} = vi$. Instantaneous rotational mechanical power is the product of the angular velocity Ω of the shaft and the torque T transmitted by the shaft, or $P_{mech} = \Omega T$. It can be shown (through a physical analysis of the motor) that

$$T = -TFi,$$

where TF is called the *transformer constant*. Due to the power conservation requirement, above,

$$\Omega = \frac{1}{TF} v.$$

Typically, however, motor manufacturers use inconsistent units such that two seperate motor constants are used. This is unnecessarily confusing, but common.

There are two obvious trade-offs: torque-current and angular velocity-voltage. But there are also power considerations: for constant power, one must trade-off angular velocity and torque. At low speeds, high torque is possible, but at high speeds, we typically are left with low torque.

Typical dc motors have unloaded angular speeds of 3000 to 9000 rpm. This is much too fast for most robotics applications, which require low speeds and high torque. Technically, we could work in this regime with high current and low voltage. *However*, there are three big problems with that:

1. this would require high current, which many circuits cannot do well;
2. high currents mean lots of heating, which can melt 💩 (heat rate $q = i^2/R$); and
3. dc motors tend to be inefficient at high current (losses to heat).

Therefore, we would like the motor to be humming along at high speed and low torque and transform that to high torque and low speed. Enter **gears**.

└ Gearing

Gears can be used to trade-off speed and torque. Gears are pretty efficient, so their transmitted power $P = \Omega T$ is approximately conserved. This means that, while the product of the angular velocity and torque is constant through the transmission, we can trade speed for torque, which is typically what we want to do in robotics applications.

Gears are **transformers** in the system dynamical sense. If the **input gear** (the one on the motor-side) has radius r_1 and number of teeth n_1, and the **output gear** has radius r_2 and number of teeth n_2, the **gear ratio** is $N = r_1/r_2 = n_1/n_2$ and the **transformer ratio** is $TF = -1/N$. This means

$$\Omega_2 = -N\Omega_1 \; T_2 \qquad = \frac{1}{N} T_1.$$

For robotics, we typically want a speed decrease and a torque increase, so, typically $r_2 > r_1$ and $N < 1$.

When using gears, we have to worry about **backlash**: the looseness or gapping between meshing gear teeth. When a gear switches direction, there is usually a chance of momentary loss of transmission. This can cause errors to accrue in positioning, so gears must be selected carefully for each application.

└ Servo motors

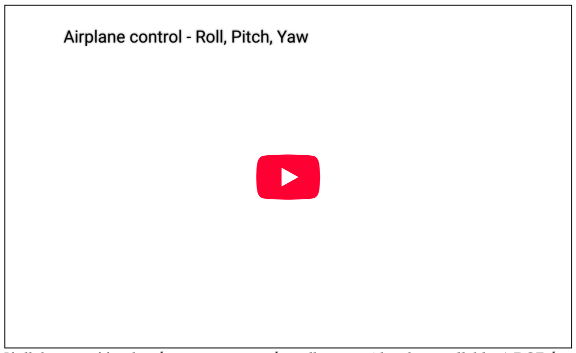
Another type of motor is very useful for robotics: **servo motors**. Actually, servo motors are just higher quality dc motors that are usually used with an angular position sensor (encoder) and a feedback controller to control the position. They are especially useful for robot arm (and other effector) positioning.

Typically, these controllers accept **pulse-width modulation** (**PWM**) signals to instruct them how much to turn.

Most robotic actuators are used to control the position of effectors. This is called **position control**. However, this can be problematic in some instances, especially if the robot is obstructed. A position controller might supply too much current to the motor, which may ruin the motor or mechanically break either the robot or something in its environment. For this reason, **torque control** is sometimes used, which is generally safer. Limiting torque limits the current (since they are proportional) and limits the mechanical loading on the robot and its environment. We might consider torque control for our course project.

Degrees of freedom

The number of **degrees of freedom** (**DOF**) for a mechanical system is equal to the size of the minimum set of coordinates required to describe the motion of a system. A mechanical system that is free to translate and rotate in all three dimensions has six degrees of freedom (three translational, three rotational). For many robots, we use the terminology for the three rotational degrees of freedom **roll**, **pitch**, and **yaw**, as shown in the video, below.



If all degrees of freedom have an actuator, then all are considered **controllable**. A DOF that does not have an actuator is considered **uncontrollable**. Obviously, a fully controllable robot is preferable.

A robot can have uncontrollable DOFs and still be effective. A car is an example of such a system. It has (in a simplified model) three DOFs and two actuators (accel/decel and turning). Yet it can reach any point in the plane with any orientation, although it may have to take a complicated **trajectory** (path) to arrive at its final configuration.

Let CDOF be the number of controllable DOFs and TDOF be the total number of DOFs in a system. There are three possibilities:

1. CDOF = TDOF. We call these robots **holonomic**. (E.g. helicopter.)
2. CDOF < TDOF. We call these robots **nonholonomic**. (E.g. car.)
3. CDOF > TDOF. We call these robots **redundant**. (E.g. robot arm)

A note on redundant robots: these have multiple trajectories to their final configuration.

Effectors are used in two different manners: for **locomotion** (moving itself around) and **manipulation** (moving other objects around). We will now consider each, in turn.

Locomotion

Here are some effectors and actuators used for locomotion:[2]

- legs,
- wheels,
- arms,
- wings, and
- flippers.

Wheeled locomotion is usually the easiest because stability is an issue with most of the others, especially legs.

Stability

A robot's **stability** is its tendency to not fall, lean, or wobble. We will consider two different types of stability: static and dynamic.

Static stability is stability when the robot is merely trying to stay still. If we are considering a ground robot, this means something like "stand still." The points of contact with the ground and the polygon (**polygon of support**) that can be traced among them are important, as is the center of mass. If the center of mass is inside the polygon, it is statically stable. One and two points of contact are statically unstable (if perturbed from balance, the robot will fall). Three points of contact gives stability because there is at least a small area to the polygon of support. More points of contact usually give greater stability. Even with three or more points of contact, the center of mass of the robot must stay above the polygon of support for stability.

A robot walking without ever becoming statically unstable is said to be performing **statically stable walking**. This requires at least four legs (or similar effectors) and tends to be slow and inefficient. This brings us to the second type of stability.

Dynamic stability is stability achieved by actively stabilizing a statically unstable system. For one-legged robots, this is called the **inverse pendulum problem**. Dynamic stability can give stability to a statically unstable system or any system that is moving about. This type of stability can be used to make walking and other forms of locomotion faster and more-efficient. It is also harder to achieve than static stability, of course.

Prototyping platforms

As described in the introduction, open-source software and hardware enable even a novice to rapidly prototype ideas. Three platforms important for prototyping will be considered here: Arduino, Raspberry Pi, and 3D printers.

Arduino

Arduino is an "open-source electronics platform" that includes open-source **microcontroller** boards and an **integrated development environment (IDE)** for programming the microcontrollers. The open-sourcedness of the boards means they can be (and routinely are) built and sold by anyone. However, Arduino does sell its own branded versions, as well.

The boards have small (relative to general purpose computers) microprocessors that can run a program over-and-over. What makes these microcontrollers useful specifically for robotics is their ability to interface with sensors and actuators.

└ Input and output pins

Pins are electrical conductors protruding from board **connectors** that can be connected to sensors and actuators. Arduino boards typically have at least one or two **analog input** pins that can be connected to analog voltage signals. Many more **digital input** pins, which can be connected to digital voltage signals, are typically available. These inputs are use to interface primarily with sensors, which can be either digital or analog. There are also **analog** and **digital output** pins that can be used to interface with, for instance, actuators.[7]

└ Power

Arduino microcontrollers can be powered by dc power supplies of many varieties: "wall-wart," computer USB, bench-top, chassis-mounted, or battery. For a typical application, low-power is typical. However, Arduinos cannot pass much power through to its output pins. Small actuators that do not require much power may be suitably powered from Arduino output pins, but many actuators used in robotics—for instance, most motors—require external power.

An external power supply can be controlled by an Arduino's outputs, however. There are several well-known circuits, such as the low- and high-side switches, that can be used to control a high-power supply with a low-power signal. Complex circuit elements such as transistors are required for these circuits. However, specialized microchips integrated into "evaluation" **printed circuit boards (PCBs)** are available at relatively low cost. For instance, a brushed dc motor can be driven with a Pololu "brushed dc motor driver."

These boards usually come with voltage-reversing capabilities that are implemented with H-bridge circuits. The inputs to the boards are the high- and low-power supply dc signals and digital signals that specify the voltage magnitude and sign.

└ Pulse-width modulation

The voltage "magnitude" just referred to for a dc motor driver is actually not a simple analog dc voltage. It is, in fact, a digital signal switched on-and-off rapidly (> 1 kHz, but faster is typically better) called a **pulse-width modulation (PWM)** signal. Let the "on" voltage be V_s. The **duty cycle** $\delta \in [0, 1]$ of the digital signal gives an average voltage $\overline{V}$ of

$$\overline{V_s} = \delta V_s.$$

This average voltage is what many systems, including dc motors, effectively "see" when a PWM signal is applied. The reason for this is that many systems do not respond as fast as the PWM signal switches, which then has a "smoothed" or averaged effect.

Another common application of PWM signals is to **light-emitting diodes (LEDs)**, which can be driven inefficiently with **voltage divider circuits** or with PWM signals. Direct application of a source voltage to an LED would nearly short-circuit the supply because LEDs have very little resistance. Adding a resistor in series with the LED limits the current but wastes power. PWM signals enable efficient LED operation.

└ Programming

The Arduino programming language is a dialect of C/C++. The Arduino IDE can be used to program the boards while connected to host computers via USB or WiFi.

Raspberry Pi

Raspberry Pis are inexpensive single-board computers that typically run variations of the Linux operating system. These are different than Arduinos in that they are full computers that can perform multiple tasks and not simply microcontrollers. Complex robotics controllers, especially those with complex user interfaces or computing tasks, are often easier to implement with Raspberry Pis.

The Raspberry Pi has many interfaces, including display and networking interfaces. However, analog input and output pins are not provided natively on the Pi. Many complex projects that require a Pi can also incorporate an accompanying Arduino.

3D printers

3D printers, now ubiquitous, are essential for rapid prototyping. Solid models of mechanical parts can be drawn in CAD software and, with very little work, can be "printed" in plastics such as ABS and in metals such as aluminum. It is important to note that, while these parts are typically still not on par with molded or machined parts, they are improving steadily. They are certainly ideal for rapid prototyping and are starting to become more widely used in production.

An introduction to PID control

PID control is introduced in this handout.

An exercise

Footnotes and References

1. This introduction generally follow Maja J. Mataric's *The Robotics Primer*, a great introductory resource for the student of robotics. ↩
2. Maja J. Mataric. *The Robotics Primer*, 2007, Massachusetts Institute of Technology. ↩ ↩2 ↩3 ↩4 ↩5 ↩6 ↩7
3. Norman S. Nise. *Control Systems Engineering*, 2011, John Wiley & Sons, Inc. ↩ ↩2 ↩3
4. Norbert Wiener. *Cybernetics: Or Control and Communication in the Animal and the Machine*, second edition, 1961 (1948), MIT Press. ↩
5. A great resource on emergence, AI, and robotics is that of Footnote 6. ↩
6. R. Pfeifer and J. Bongard. *How the Body Shapes the Way We Think: A New View of Intelligence*, 2006, MIT Press. ↩
7. Often a digital pin can be configured as either input or output. ↩

Rico A.R. Picone, PhD
rico@stmartin.edu

Dr. Rico to you

Page updated 13 July 2021 at 16:54