Using/Coding Single Board Computers and Microcontrollers to Overcome Simulation Challenges

#SIMOPS2020

WELCOME

Myles Larson

Simulation Operations and Technology Coordinator
North Idaho College, Coeur d'Alene, Idaho
myles.larson@nic.edu

Disclosure:

Sole proprietor:
sim4d.healthcare, LLC

Modifying Your Manikin:

If you choose to modify your manikin or other hardware that is under warranty, check with your manufacturer or warranty provider first. I am not responsible for voided warranties.

Learning Objectives:

Objective 1: Identify the purpose of the device or modification and select the appropriate SBC or microcontroller to best meet the needs of the project.

Objective 2: Implement an online integrated development environment (IDE), use GitHub repositories, and execute hardware add-on specific libraries as resources for code examples and tutorials to aid in writing software for the selected project and hardware.

Objective 3: Examine basics of editing and compiling code, and programming the selected hardware.

Getting Started:

Integrated Development Environment (IDE), Libraries, and Additional Software

Log in to the Arduino Online IDE: https://create.arduino.cc/

Open GitHub Respository: https://github.com/myles-larson

Install Arduino Agent (optional plug-in for this workshop but needed for interfacing with Arduino hardware): https://create.arduino.cc/getting-started/plugin/welcome

Single Board Computers (SBCs) and Microcontrollers:

What is a single board computer (SBC)?

What is a microcontroller?

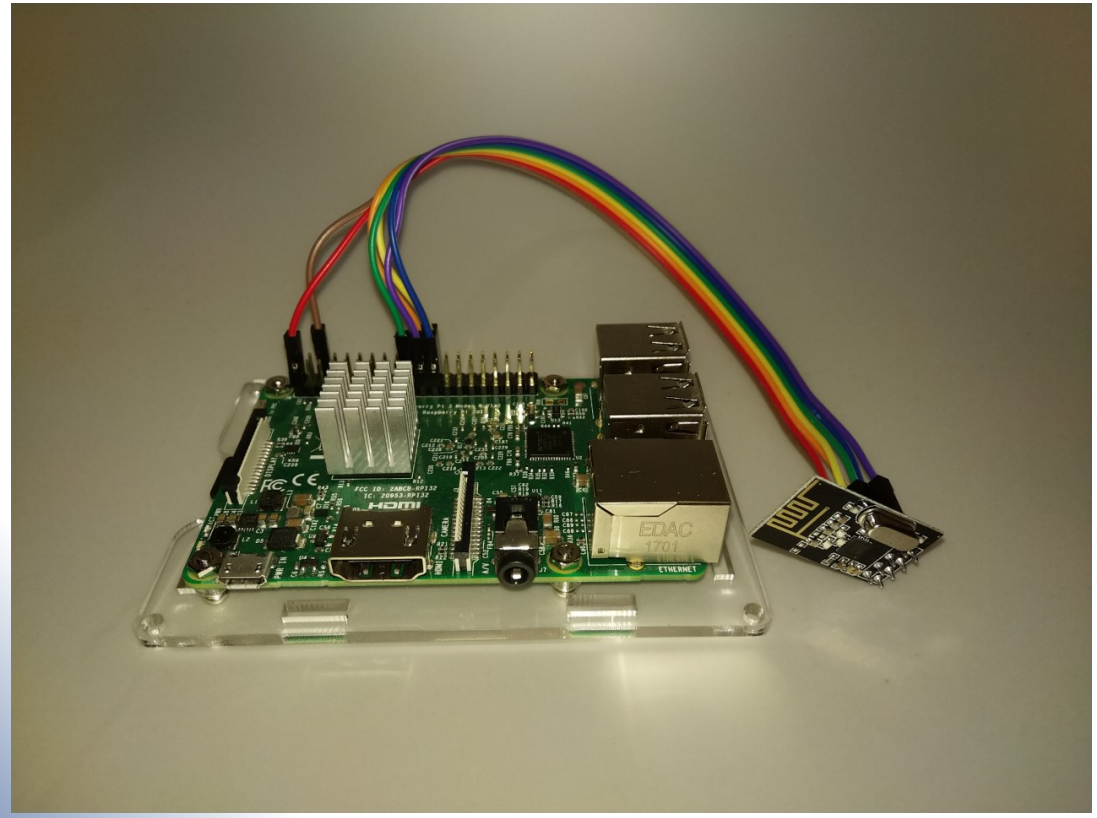
Why choose one over the other?

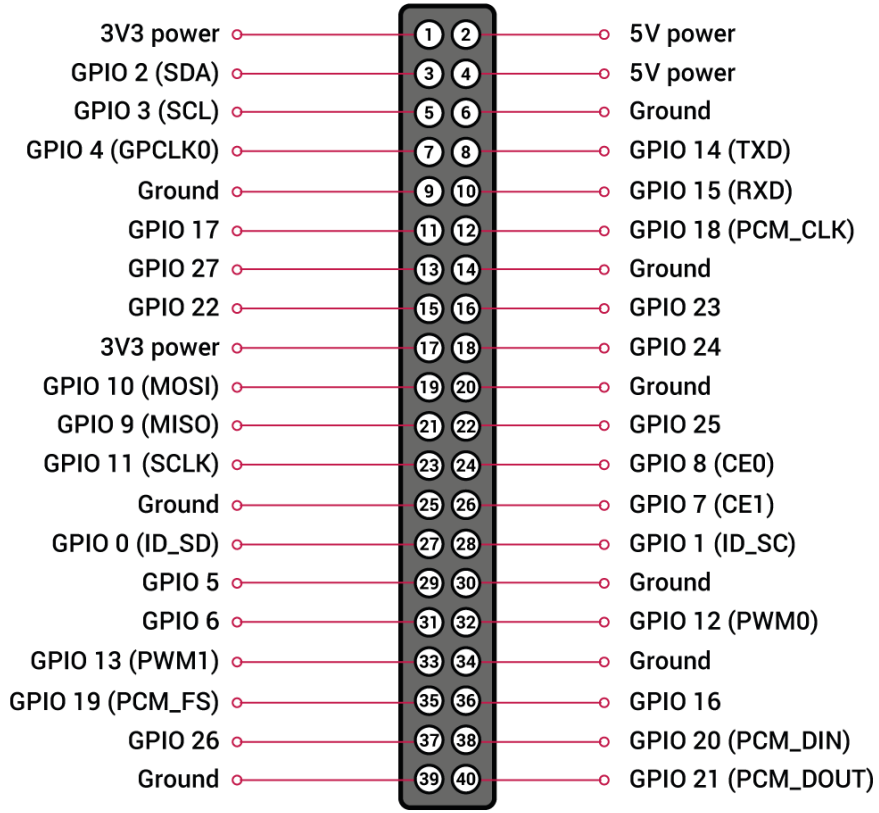
General Vocabulary:

- **GPIO:** General Purpose Input/Output. The pins we use to connect and control hardware.
- **Pins:** Digital, analog, power, and ground. Depending on the platform, the type and number of pins will vary.
- **SBC**: Single Board Computer. A self-contained computer with all required components (except sometimes storage) on one circuit board.
- **Microcontroller:** A single chip with processor, memory, and storage integrated. Can be programmed to run a variety of commands with inputs and outputs.
- **Memory:** Fast, short-term hardware for accessing and manipulating data. Usually volatile, meaning it "forgets" what it was storing when power is removed.
- **Storage:** Slower, long-term hardware to store and access data. Is non-volatile, meaning it keeps what it was storing after power is removed.
- **Shields and HATs:** Add-on boards designed to easily work with the selected platform. Shields work with Arduino and HATs (sometimes pHATs or wHATs) work with Raspberry Pi. HAT is an acronym for "Hardware Attached on Top."
- **Breadboard:** An electronics prototyping board designed for easily building circuits without soldering.
- **Physical Computing:** Using SBCs and microcontrollers to manipulate physical components such as LEDs, motors, and valves.
- **IDE:** Integrated Development Environment. Software designed for programming. Can be specific to a programming language, to a hardware set, or can be expanded to support multiple platforms through add-ons.
- **Functions:** a named section of code that performs a task.
- **Calls:** Using the name of the function to invoke the task it performs.
- **Arguments:** Some data that the function requires to perform its task.
- **Pass:** The term used to describe what happens to the arguments. We pass arguments to a function when the function is called.
- **Return:** The output we expect back after the function has run. This may be "void," meaning the function does not "return" anything.
- **SPI:** Serial Peripheral Interface. 5-6 pin (plus power and ground) standard for serial communication between hardware devices such as SBCs and microcontrollers.
- **I2C:** Inter-Integrated Circuit. 2 pin (plus power and ground) serial protocol for communication between hardware devices such as SBCs and microcontrollers.

Raspberry Pi 3 B+:

SBC with 40 pins (28 GPIO) running a Linux (and Windows, sort of) operating system





Raspberry Pi 3 B+:

SBC with 40 pins (28 GPIO) running a Linux (and Windows, sort of) operating system

Pros:

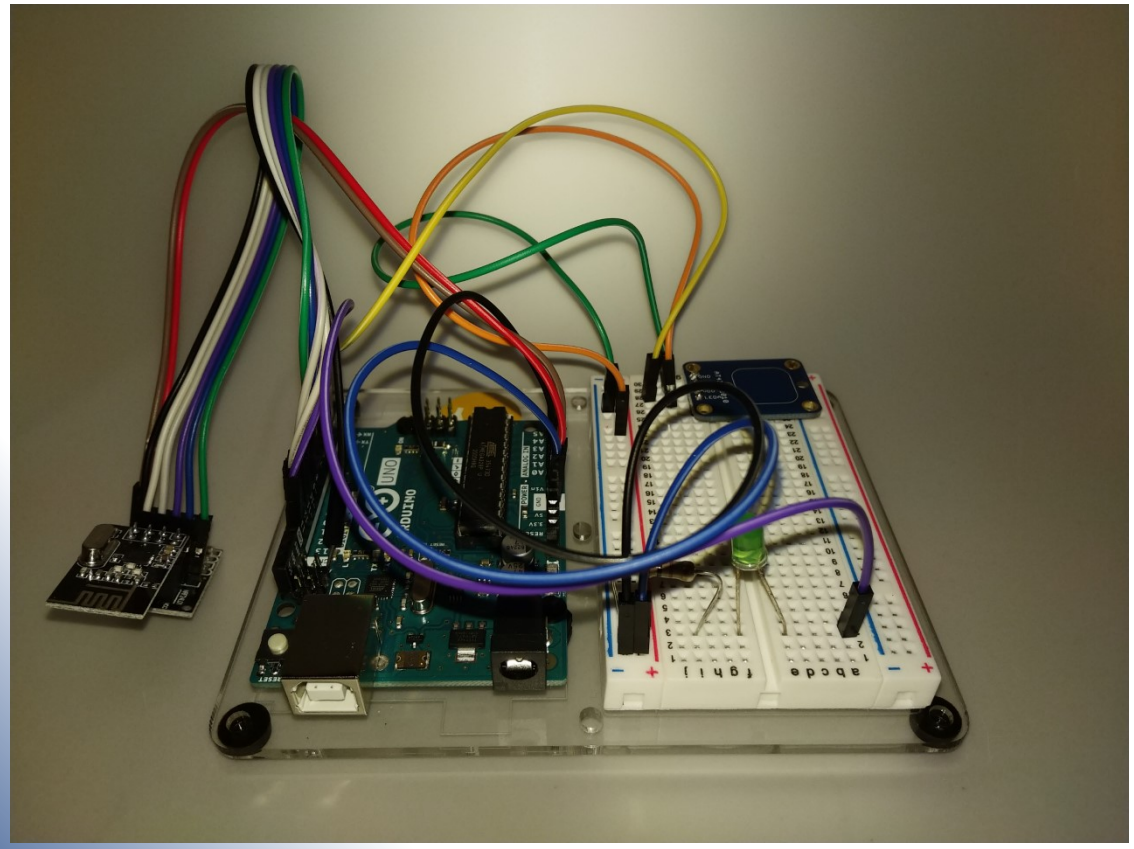
- Can write code directly on the device, no separate computer required
- More memory and faster processors than most microcontrollers
- Expandable storage
- Multiple programming languages supported
- On-board networking, video, and peripheral ports

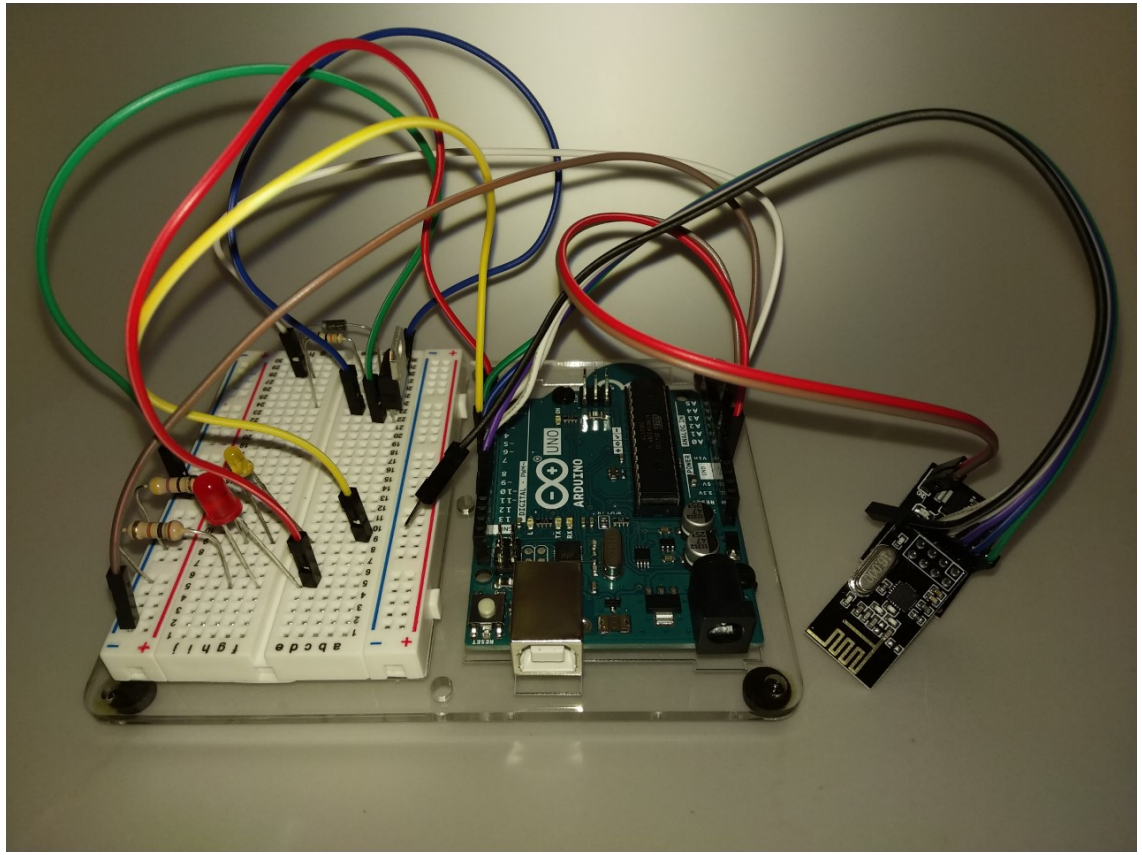
Cons:

- Operating system uses some memory and storage
- Higher power consumption
- Fewer power supply options
- Only 5V and 3.3V power output
- Only reads digital inputs, no analog input support
- Power failure can corrupt storage

Arduino Uno Rev3:

Motherboard for the ATmega328P microcontroller with 14 digital and 6 analog pins





Arduino Uno Rev3:

Motherboard for the ATmega328P microcontroller with 14 digital and 6 analog pins

Pros:

- Low power consumption
- Three power supply options
- Digital and analog pins
- 5V, 3.3V, and power supply determined power output (up to 12V)
- Fault tolerant to power failure (will restart)
- Lower cost

Cons:

- One programming language supported
- Limited memory and storage (about 2KB and 32KB respectively)
- No on-board networking, video, or peripheral ports

Arduino Uno vs. Raspberry Pi 3:
What are they good for?

Arduino:

- Simple output from sensors
- Analog sensors
- Simplicity in connecting to and communicating with other device components

Raspberry Pi:

- Network/Internet connected projects (WiFi, Ethernet, and Bluetooth)
- Multimedia
- Multitasking

Project Ideas and Which Platform to Choose:

Manikin missing a feature?

Too much pretending involved with some real devices?

Only need a few features (and/or lower $) of the real thing?

Project Ideas and Which Platform to Choose:

Brainstorming

Example Projects:

Let’s look at some code:
https://github.com/myles-larson/SimOps2020

Online Resources:

GitHub and Git:
https://github.com/myles-larson/SimOps2020/tree/Arduino

Arduino IDE:
https://create.arduino.cc/editor/

RadioHead Library:
https://www.airspayce.com/mikem/arduino/RadioHead/classRH__NRF24.html

Arduino Online IDE:

Let’s write some code.

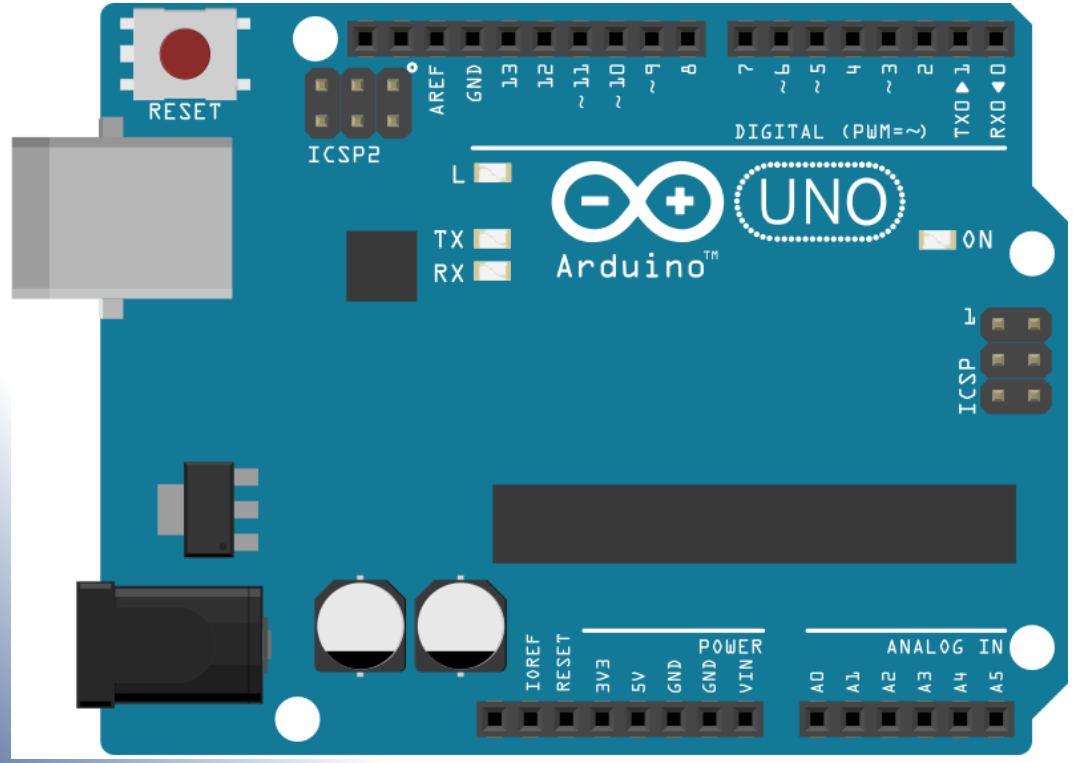
Compile code with errors and correct them:
Let’s see how the IDE helps us find and correct errors.

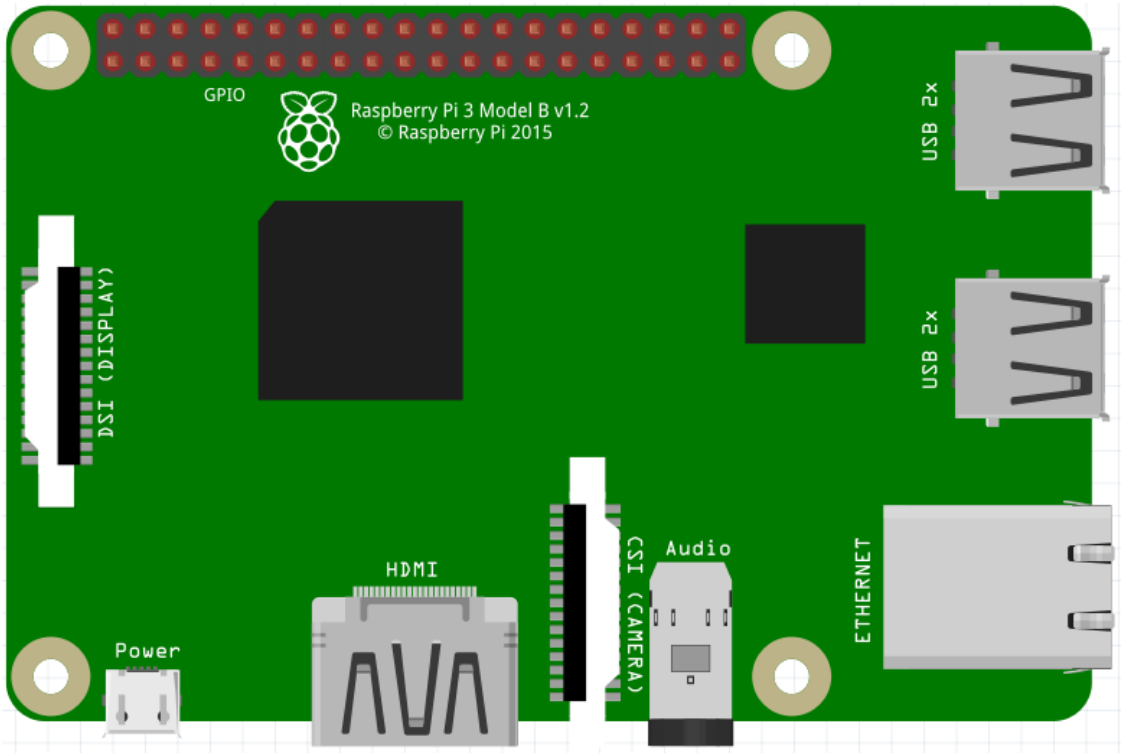
Make an LED flash:
Start a new Arduino project and write our first sketch.

A step further, make an LED fade:
Instead of flashing, how can we make the LED fade?

Challenge Question:

How many <u>more</u> digital GPIO pins does a Raspberry Pi have than an Arduino Uno?





Challenge Question:
Answer

How many more digital pins does a Raspberry Pi have than an Arduino Uno?

14 more pins.
Raspberry Pi has 28 digital GPIO pins and the Arduino Uno has 14.

Compile and Run:

Let’s compile our code, upload it to a pair of Arduino’s, and see if it works.

Additional Resources:

- Programming Arduino, second edition, by Simon Monk.
- Circuit.io, hardware and circuit planning: https://www.circuito.io/
- Adafruit, hardware and tutorials: https://www.adafruit.com/
- Spark Fun, hardware and tutorials: https://www.sparkfun.com/
- Pinout!, GPIO pinout guide for Raspberry Pi: https://pinout.xyz/
- Arduino, hardware and tutorials for the Arduino family of boards: https://www.arduino.cc/
- Raspberry Pi Foundation, hardware and tutorials for the Raspberry Pi family of boards: https://www.raspberrypi.org/
- Instructables, project ideas, instructions, code, and hardware listings: https://www.instructables.com/
- GitHub, online platform to learn, share, create, and collaborate to build software: https://github.com/
- DigiKey, inexpensive hardware supplier: https://www.digikey.com/

QUESTIONS?

#SIMOPS2020

Thank you.
Using/Coding Single Board Computers and Microcontrollers to Overcome Simulation Challenges

#SIMOPS2020