CP316
High Level and Low Level Programming

Terry Sturtevant

Wilfrid Laurier University

June 27, 2019

Arduino origin

Arduino origin

The Arduino was developed for use by *non-technical* people;

Arduino origin

The Arduino was developed for use by *non-technical* people; artists, hobbyists, etc.

Arduino origin

The Arduino was developed for use by *non-technical* people; artists, hobbyists, etc.

The software is simplified from C/C++ to make it easier to learn.

Arduino origin

The Arduino was developed for use by *non-technical* people; artists, hobbyists, etc.

The software is simplified from C/C++ to make it easier to learn.

This is good for people without knowledge of the hardware.

Arduino origin

The Arduino was developed for use by *non-technical* people; artists, hobbyists, etc.

The software is simplified from C/C++ to make it easier to learn.

This is good for people without knowledge of the hardware.

However, it is *possible* to do **low-level** programming as well.

High or Low?

High or Low?

- **High level** programming allows your program to run on *different* devices.

High or Low?

- **High level** programming allows your program to run on *different* devices.

  Low level programming allows your program to use features of *specific* devices.

High or Low?

- **High level** programming allows your program to run on *different* devices.

  Low level programming allows your program to use features of *specific* devices.
- **High level** programming *allows you to ignore* details of underlying hardware.

High or Low?

- **High level** programming allows your program to run on *different* devices.

  Low level programming allows your program to use features of *specific* devices.
- **High level** programming *allows you to ignore* details of underlying hardware.

  Low level programming *allows you to control* details of underlying hardware.

Principles

Principles

- **High level** programming is much easier.

Principles

- **High level** programming is much easier.

  It's a better choice when hardware and timing are not critical.

Principles

- **High level** programming is much easier.

  It's a better choice when hardware and timing are not critical.

  Using built-in libraries saves lots of time and effort.

Principles

- **High level** programming is much easier.

  It's a better choice when hardware and timing are not critical.

  Using built-in libraries saves lots of time and effort.
- **Low level** programming allows precise control of timing.

Principles

- **High level** programming is much easier.

  It's a better choice when hardware and timing are not critical.

  Using built-in libraries saves lots of time and effort.
- **Low level** programming allows precise control of timing.

  Usually only a few small sections of code are timing-critical.

Principles

- **High level** programming is much easier.

  It's a better choice when hardware and timing are not critical.

  Using built-in libraries saves lots of time and effort.
- **Low level** programming allows precise control of timing.

  Usually only a few small sections of code are timing-critical.

  Allowing as much processing as possible at the high level simplifies coding.