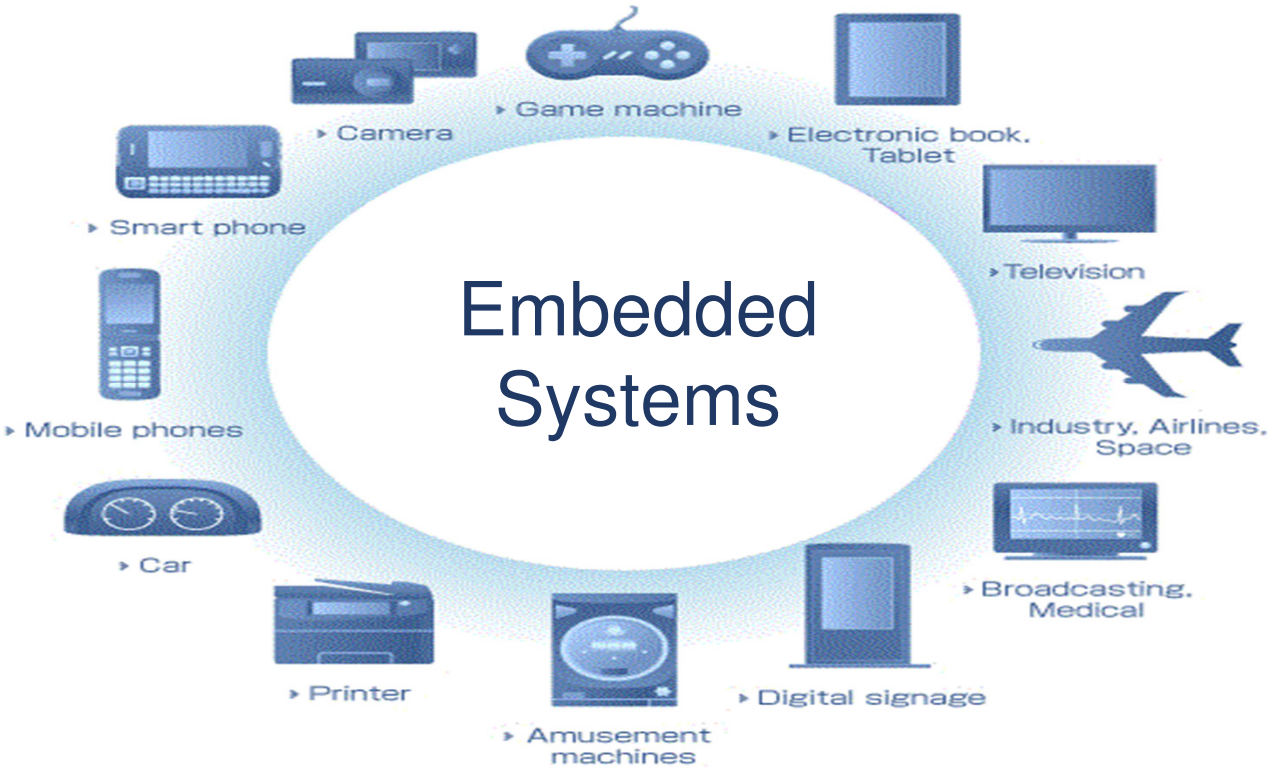
SoRTES Lab - Session 1

Ashok Thangarajan, Stefanos Peros, Emekcan Aras

DistriNet

What are these?



DistriNet

Embedded Computing

- Developed for a specific purpose
- Resource constrained environment

General Purpose Computing

- Wide range of tasks
- Unlimited resources (!= embedded world)
- Fast processor speed (!= embedded world)

Comparison

	Embedded Computing	General Purpose Computing
Processor Speed	10 - 800 MHz	Several GHz
Memory	Few KB to MB	Several GB's
Operating system	RTOS – various embedded OS	General purpose OS
Objective	Specific purpose	General purpose computing

Device classes

- IETF classification

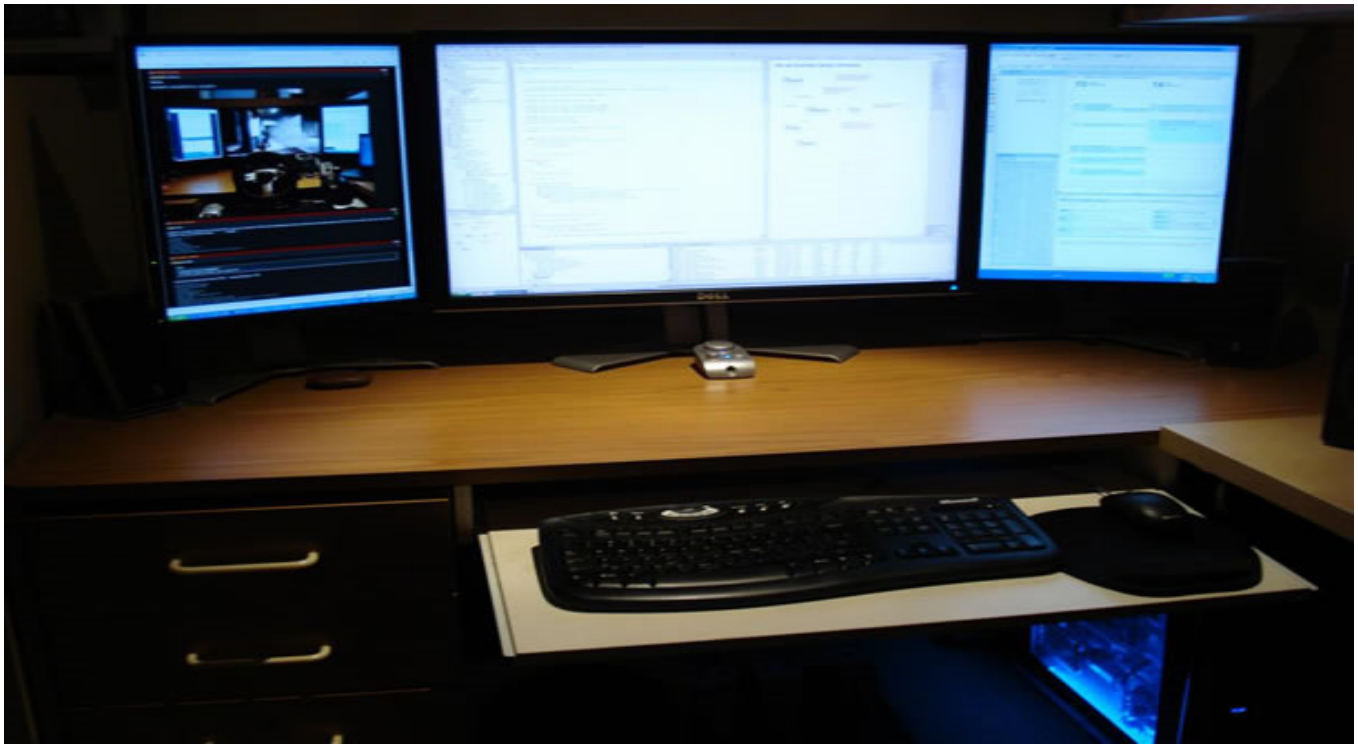
Name	data size (e.g., RAM)	code size (e.g., Flash)
Class 0, C0	<< 10 KiB	<< 100 KiB
Class 1, C1	~ 10 KiB	~ 100 KiB
Class 2, C2	~ 50 KiB	~ 250 KiB

- Arduinos are Class 0 devices (SRAM – 2KB, Flash – 32 KB)
- Rpis are Class 2 devices

Embedded Programming

How do we program this?

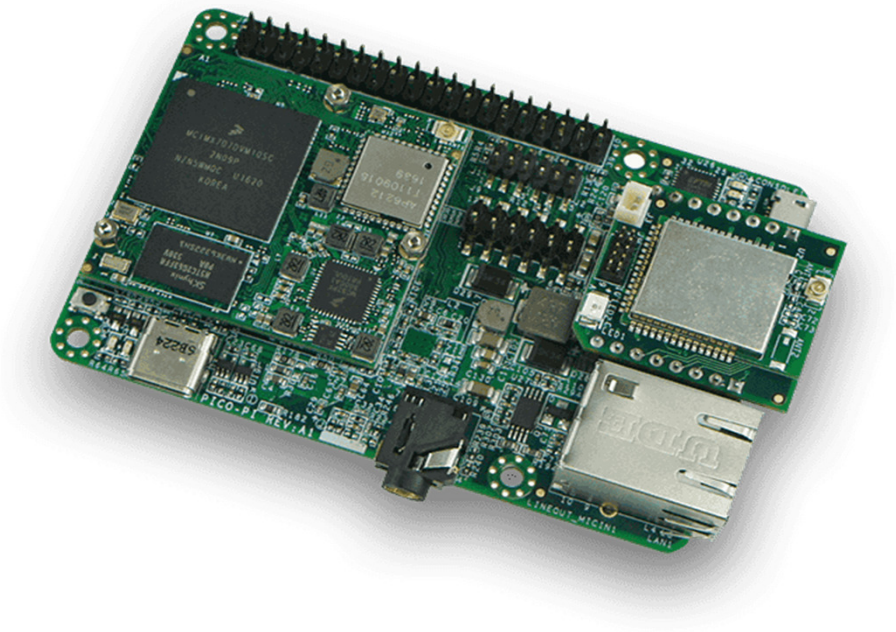
- General purpose application development



DistriNet

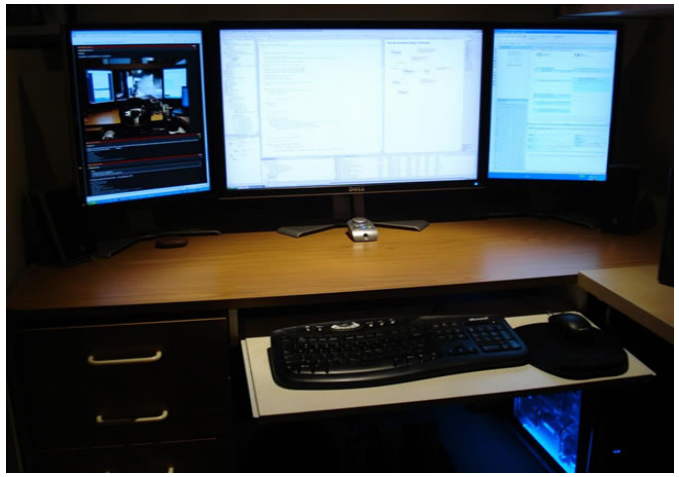
How do we program this?

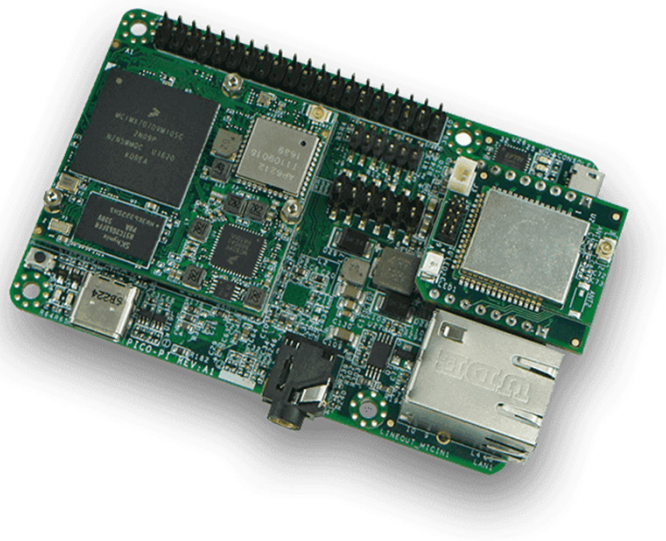
- Typical embedded development space



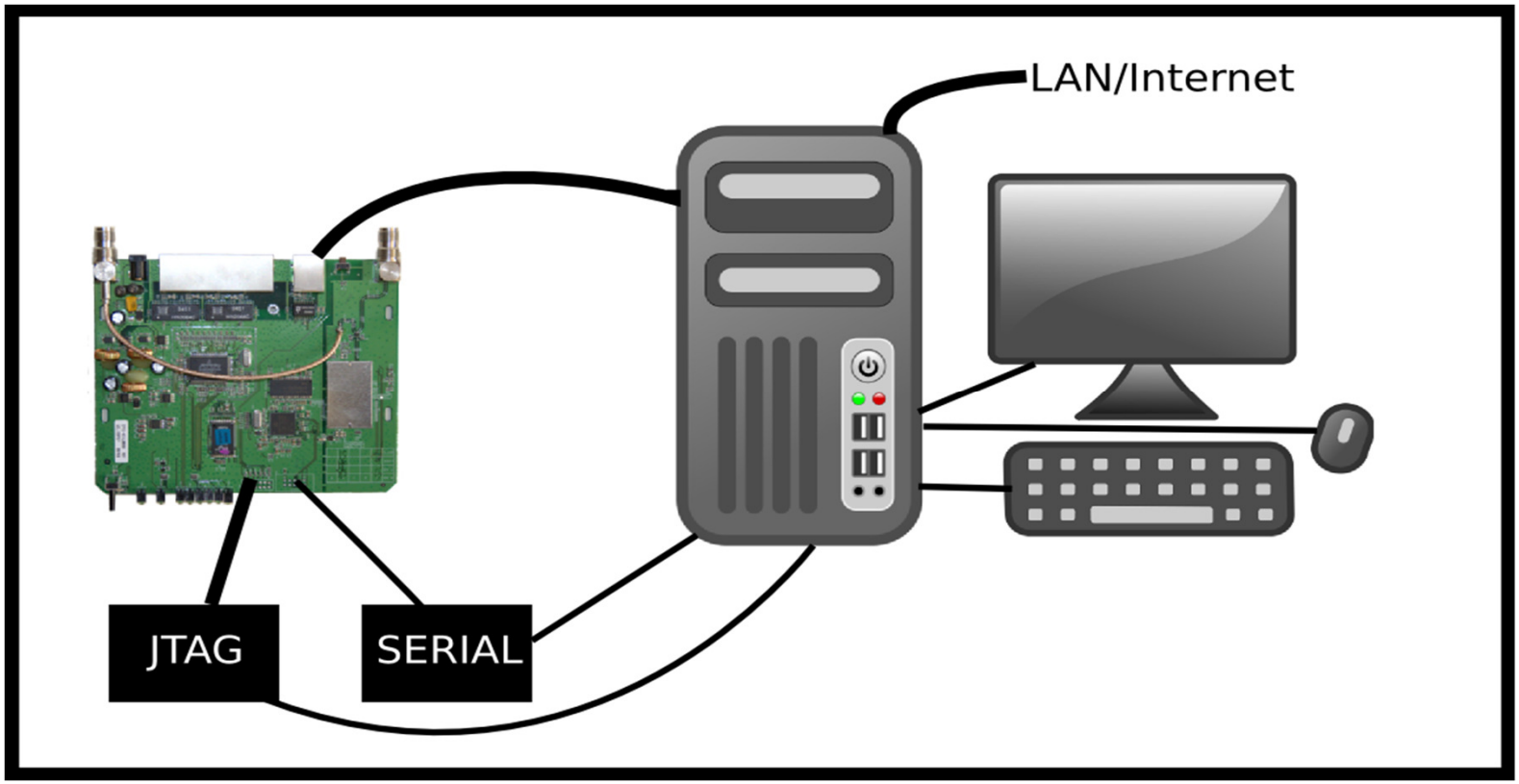
How do we program this?

- General purpose application development → Embedded Programming





Develop in a host machine

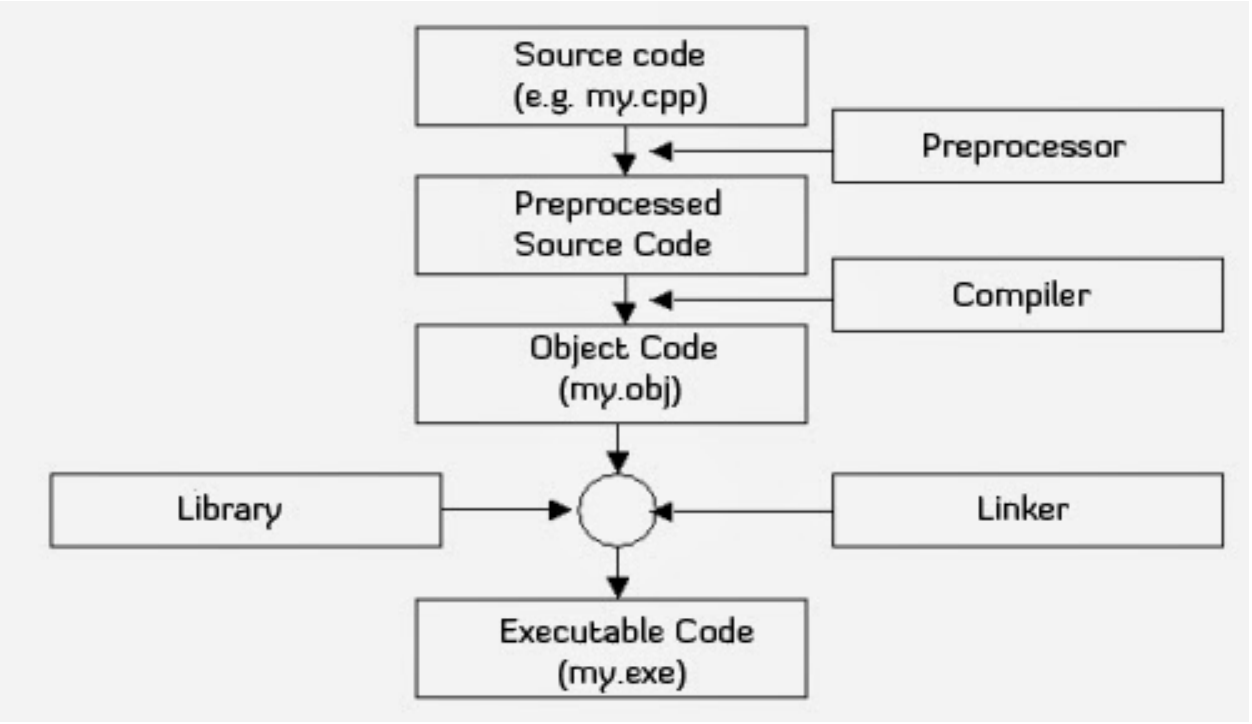


DistriNet

Tool Chain

- Cross Compiler
  - To compile a program for another machine architecture
- Assembler
  - Assembly to object code
- Linker
  - Linking many object code and libraries to an executable or library.
- Debugger
  - Interface to debug programs, generally with external hardware support
- Other helper binaries
  - Archives, nm, readelf, etc..

Typical compilation phases in a compiler



DistriNet

Programming Interface

- Multiple programming interfaces
- Common ones are:
  - JTAG
  - SPI/UART
  - USB
  - Arduino serial bootloader

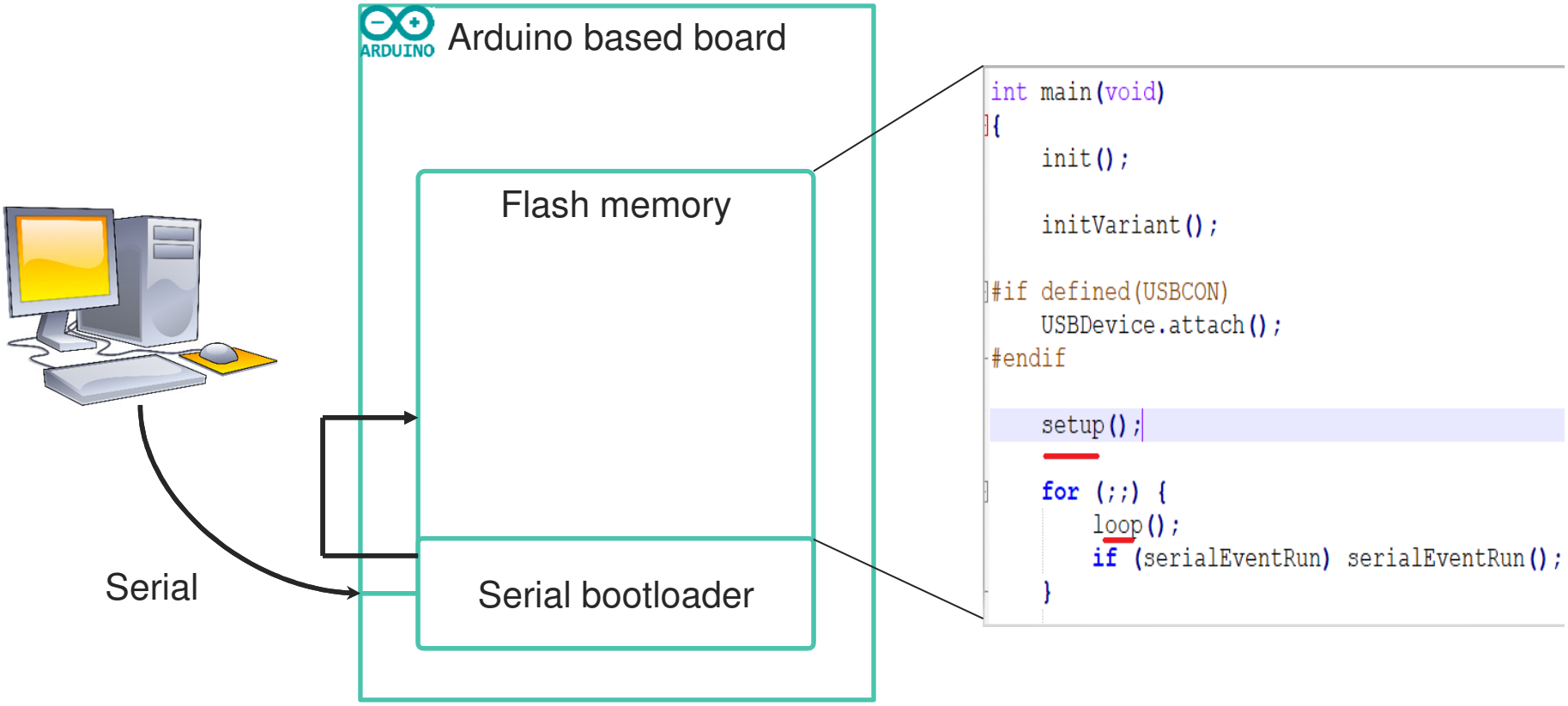
Boot-up Sequence

- Initialize processor
- Done by processor-specific boot sequence, generalized to:
  - Power on Reset
  - Initial State
  - Bootloader starts running
  - Bootstraps the operating system (OS)
  - OS starts up and completes boot up sequence
  - Starts the application
  - Wait to load program on **programming interface**

Boot-up Sequence

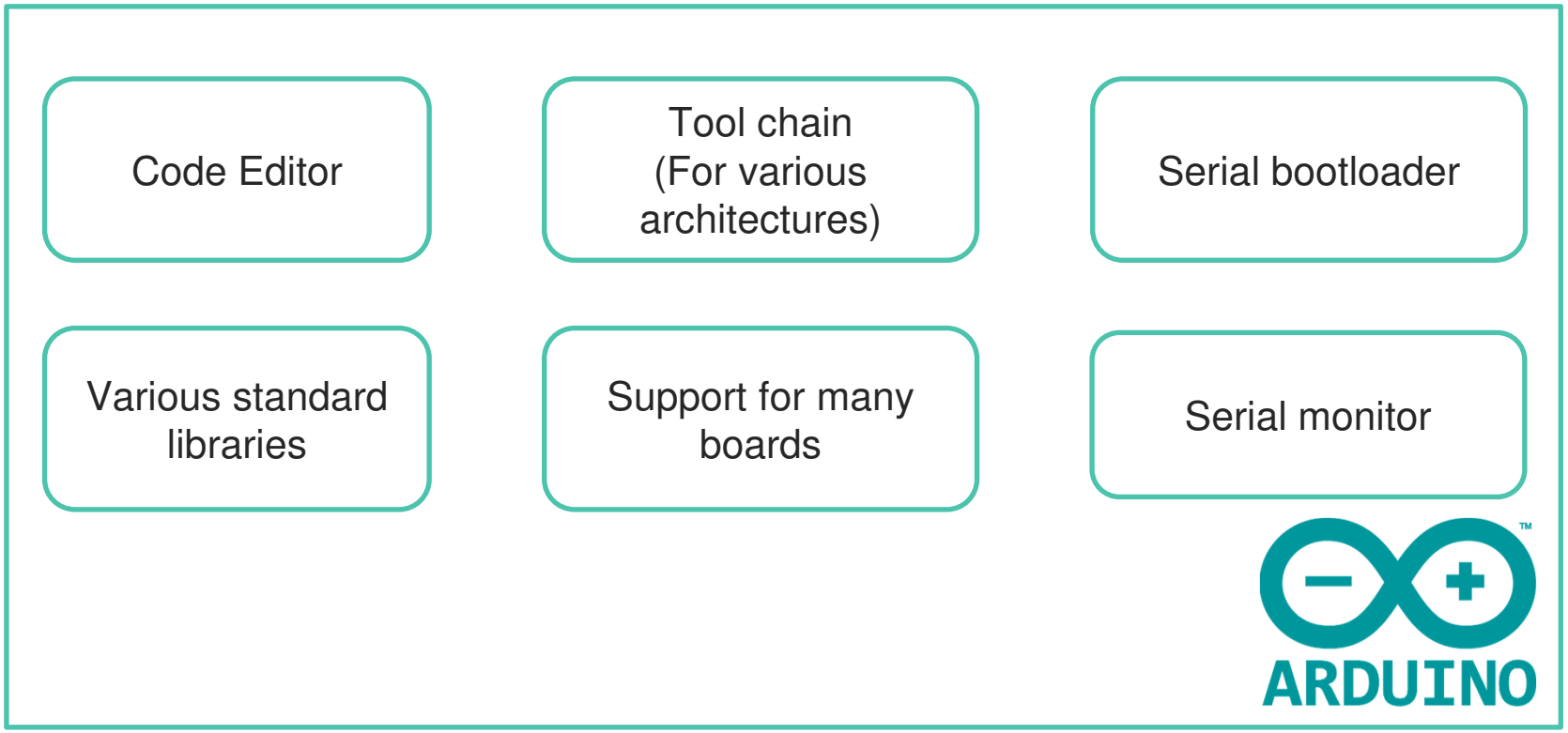
- Some processors boot sequence is simple
- After reset vector, starts executing code at 0x0000
- Many advanced controllers and embedded processors have complex boot sequences

Arduino Bootloader



DistriNet

Arduino IDE


Code Editor
Tool chain (For various architectures)
Serial bootloader
Various standard libraries
Support for many boards
Serial monitor
ARDUINO

Arduino code structure

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Example code

```
int pushButton = 2;

void setup() {
    Serial.begin(9600);
    pinMode(pushButton, INPUT);
}

void loop() {
  int buttonState = digitalRead(pushButton);

  Serial.println(buttonState);
  delay(1);
}
```

Why and why not Arduino?

- Why
  - Easy to learn
  - Focus on the application aspects
  - Rapid prototyping

- Why not?
  - Libraries not optimized
  - Non-functional requirements of product may not be fulfilled
    - Meeting industry standards for industrial products
    - Meeting safety standards for safety related products

Data Sheet

- Sometimes we have to read a Digital IO
  - How do we know if a pin is an IO pin or it is exposing an internal peripheral?
  - I2C timing characteristics?
  - How much current an IO pin can drive?

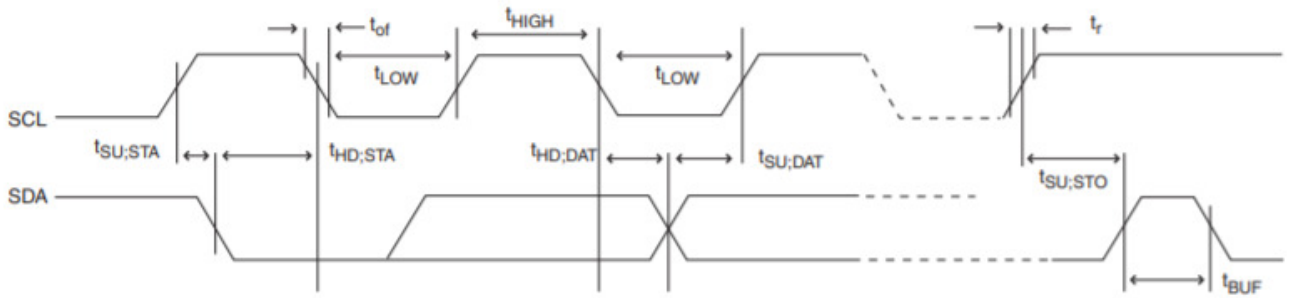
Data Sheet – from AtMega32U4 Datasheet

- Current per IO pin

DC Current per I/O Pin 40.0mA

- I2C timing characteristics

Figure 29-3. 2-wire Serial Bus Timing



DistriNet

Data Sheet

- Datasheet and Technical specifications provide this info
- This is important due to the heterogeneity of embedded domain
- We may program various machine architecture, and wide range of microcontrollers with varied peripherals.

Sensors

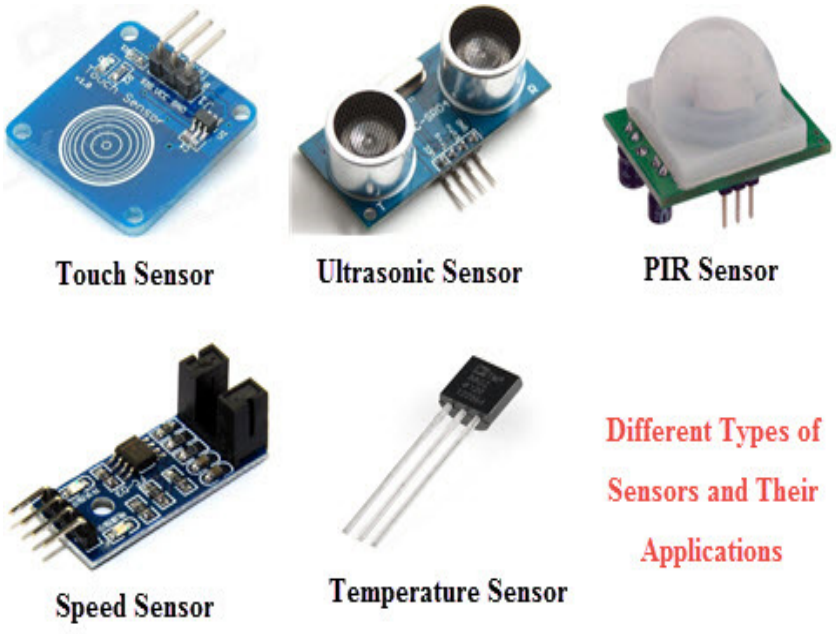
Sensors – (1/2)

- Microcontrollers need digital inputs from environment
- Sensing changes in physical elements – Sensors
- Sensors sense changes in environment

Sensors – (2/2)

- Sensor output to microcontrollers:
  - I2C/SPI/UART
  - Analog output to processors ADC
  - Digital output via IO pins

Some sensors



Arduino Sensors

Industrial Sensors

DistriNet

Calibration – (1/2)

- Sensors need calibration
- Sensors lose calibration because of various factors

Calibration – (2/2)

- Use a known quantity
- Measure it using the sensor
- Adjust for variations
- Eg: Weighing machines

Libraries for the board

- Refer to the link below
- https://github.com/BSFrance/BSFrance-avr
- How to set up board and flash
- How to set up for power measurement

Exercises

Exercise 1

- Display the following in serial port (console), from your Arduino application
  - “Enter LED status (on/off):”
- Read the user input value from serial port into your application
- If value is “on”, display the following in serial port from your Arduino application
  - “Enter the blink rate (1-60 sec):”
- Print user provided values to serial port
  - “You have selected LED on/off. Blink rate is xx sec”
- After the user selected values are displayed in serial port, the blink pattern should be seen on the onboard LED
  - Info: Use a timer library to set the blink rate

Exercise 2

- Create a timestamp module (A simple counter should be enough)
- Create a small database (You may use Arduino library)
- Calibrate and read the temperature value from Arduino and store it into the database
- When the user pressed "p" on the console, your application should print all the stored values until that point.

Thank you!

https://distrinet.cs.kuleuven.be/